UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN
TELEMÁTICA

PROYECTO FINAL DE CARRERA

Construcción práctica de una plataforma de bots que emulan ser usuarios de Twitter para apoyar el análisis del comportamiento humano en las redes sociales

AUTORA: Paloma Pérez Yagüe
TUTOR: Luis Sánchez Fernández
DIRECTORA: María Luz Congosto Martínez

27 de julio de 2011

A todas aquellas personas que han hecho posible que este proyecto salga adelante. En especial a Mariluz Congosto, por su gran labor de tutorización a lo largo de todo el proyecto, incluidas sus correcciones a altas horas de la madrugada, y su inspiración a la hora de haber propuesto un proyecto de estas características, ha sido un verdadero placer haberla elegido como tutora de proyecto, a mi familia por haber aguantado mis continuos cambios de humor en los momentos finales del proyecto, y por supuesto a Amadeo, que no se ha separado nunca de mi lado y me ha animado siempre a seguir adelante con todo lo que me he propuesto demostrándome su ayuda y su apoyo incondicional.

Gracias.

Resumen

Gracias al uso cada vez más extendido de las nuevas tecnologías y la revolución en el sector de las comunicaciones que supuso la nueva entrada en Internet de la información 2.0, son cada vez más comunes y populares aquellas plataformas sociales que confieren un especial protagonismo al usuario y le permiten mostrarse libremente ante el resto de usuarios que hacen uso de las mismas.

Por ello, en la última década se ha visto cada vez más reforzada la presencia del usuario en las denominadas redes sociales, que no son más que comunidades virtuales de individuos que tienen su nexo de conexión en intereses u objetivos comunes.

Y si hay una red social cada vez más popular en el mundo de la comunicación social, esa es Twitter, que suma un elevado número de usuarios cada día y cuya información vertida por cada uno de sus participantes, en ocasiones, tiene grandes repercusiones sociales.

Pero, ¿Qué motivo lleva a un usuario a integrarse en este tipo de redes sociales? ¿Qué tipo de relaciones pueden darse entre sus usuarios?, ¿Cuál es el patrón común de comportamiento en este tipo de redes?, ¿Qué llevan a una opinión vertida por un usuario a tener popularidad e incluso consecuencias sociales que despiertan el interés de cientos o miles de personas en todo el mundo?

Estas y otras preguntas intentan ser contestadas cada día mediante el estudio de los diferentes patrones de comportamiento que pueden ser encontrados en dicha red social, pero ¿qué mejor forma existe de observarlos que intentar recrearlos nosotros mismos y poder ver desde sus inicios la evolución social de un determinado individuo?

Abstract

Due to the extensive use of new IT technologies derived from the adoption of the Web 2.0 paradigm, there is an expanding acceptance of user centric social applications that empowers the user to communicate in a much more personal way than using conventional channels.

This has lead during the last decade to an increasing numbers of users using social networks, which are just virtual communities of people with common interests and goals.

Twitter has a growing presence among the social networks. It has a huge daily number of participants, and the information that its users interchange has, on occasions, roused great social repercussions.

But, why does a user join this kind of social networks?, What kind of relations can emerge among social network users?, Which is the common behaviour pattern in this kind of social networks?, What makes the opinion of a particular user to be popular? And, what is more, What makes it interesting for hundreds of thousands of people all around the world, even creating social movements?

These and many other questions are examined every day, analysing the different behaviour patterns found in the aforementioned social network, but what better tool to study all these matters than recreate these patterns applying a virtual user and auditing the social evolution of the virtual user from its birth?

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

> "El ser humano es un ser social por naturaleza, y el insocial por naturaleza y no por azar o es mal humano o más que humano... La sociedad es por naturaleza y anterior al individuo... el que no puede vivir en sociedad, o no necesita nada por su propia suficiencia, no es miembro de la sociedad, sino una bestia o un dios."
>
> *Aristóteles, Política*

1.1. Motivación del proyecto

El ser humano es un animal social y como tal va estableciendo lazos sociales durante toda su vida. Estos enlaces nacen en el seno de familia y se van ampliando con las relaciones de vecindad, de amistad, del colegio y del trabajo.

Dunbar estableció que de acuerdo con el tamaño del cerebro humano el número de relaciones sociales controlables por cada individuo eran 150 [Wik11]. No obstante, con el nacimiento de las redes sociales en Internet han surgido un nuevo tipo de relaciones virtuales, con vínculos más débiles, que superan ampliamente dicho número.

Es indudable que las nuevas tecnologías han potenciado la capacidad de establecer nuevas conexiones entre individuos más allá de las limitaciones geográficas y temporales. Esto ha permitido que los individuos generen identidades digitales y que desarrollen una red de contactos compuesta de individuos que se conocen físicamente y otros que permanecen en el plano virtual. En este nuevo paradigma afloran una serie de preguntas a las que este proyecto intenta aportar luz.

¿Qué necesidad sienten los individuos para conectar con personas que no conocen?

¿Qué criterios les lleva a establecer este tipo de enlaces?

¿Qué tipo de conexiones se establecen entre individuos desconocidos?

¿Qué duración tienen estas conexiones?

Una herramienta de apoyo a la experimentación social en las redes sociales es la construcción de individuos digitales robotizados (bots) que emulen el comportamiento humano con distintos patrones de comportamiento y preferencias. Esta tarea, no trivial, de emular automáticamente identidades digitales configurables facilita una metodología de análisis del desarrollo de las relaciones en las redes sociales.

De las redes sociales existentes hoy día se ha seleccionado Twitter por disponer de un API muy completo y bien documentado que hace posible la creación de bots con un esfuerzo razonable. Adicionalmente Twitter está en plena etapa de expansión de usuarios lo que permite crear experimentos a corto y medio plazo con una masa cada vez mayor de identidades digitales.

Al delimitar el entorno de los bots a la red social Twitter su aceptación social vendrá medida por el tamaño de la red que sea capaz de generar y por la repercusión de sus mensajes en dicha red. En ningún momento los bots podrán tener comportamientos inadecuados o agresivos por estar limitados todos sus parámetros dentro de los rangos culturales aceptados implícitamente por los usuarios de Twitter.

Adicionalmente a la motivación principal de este proyecto que es el apoyo a la investigación social en Twitter es posible adivinar otros usos como:

- Complemento a la actividad de un usuario: Los bots podrían realizar un cierto tipo de actividades rutinarias como la publicación de información interesante, automatizar la cortesía, o seguir automáticamente a otros usuarios.
- Ayuda a la selección de RR.HH.: Ciertas configuraciones de los bots facilitarían el descubrimiento de los usuarios más relevantes en un tema, lo que sería de ayuda a la selección de personal de ciertas empresas.
- Escucha activa: Los bots se podrían parametrizar para poder escuchar las conversaciones asociadas a un determinado tema o a una marca concreta.

1.2. Objetivos

El objetivo principal de este proyecto consistirá en la creacción de un bot con apariencia humana, con la finalidad de poder analizar el valor que se le da al comportamiento de los usuarios en Twitter, y de este modo, utilizarlo posteriormente para obtener el ***Timeline*** de los usuarios que sigue.

Para conseguirlo, se plantean una serie de objetivos parciales:

1. **Definir perfiles de comportamiento**: Se definirán distintos perfiles de comportamiento por cada bot. Para ello, se realizará un estudio de aquellas propiedades que permitan caracterizar el comportamiento de un bot en Twitter.
   Dichas propiedades se almacenarán en distintas tablas en una Base de Datos y podrán

cambiar de acuerdo a la evolución del comportamiento del bot, o mediante petición expresa por parte del usuario que lo crea.

2. **Definir estrategias de comportamiento**: De acuerdo a las características de comportamiento que serán definidas por cada bot en su perfil de comportamiento, cada bot tendrá que ser capaz de comunicarse en la plataforma Twitter e interrelacionarse con el resto de los usuarios reales.
   Para ello, se definirán distintas estrategias de comportamiento para cada una de las funciones que un usuario real podría usar en la plataforma Twitter, tales como, la publicación de mensajes, el seguimiento de nuevos usuarios, contestar mensajes privados, muestras de agradecimiento, etc.

3. **Definir el modelo de datos**: Puesto que la información del perfil del usuario y las estrategias de comportamiento del bot se almacenarán de forma persistente en Base de Datos, es necesaria una correcta definición del modelo de datos.

4. **Crear la interfaz gráfica**: Se realizará una interfaz gráfica mediante una plataforma Web, que podrá ser accedida por distintos usuarios, y que permitirá la creacción y modificación de los bots registrados en la misma. Del mismo modo, se podrá dar de baja la monitorización de los bots pertenecientes a un determinado usuario.

1.3. Contenido de la memoria

A continuación se detalla la distribución del presente documento

Capítulo 1. Introducción: En este capítulo se da una justificación del porqué de la realización del proyecto. Además de un acercamiento a los objetivos fundamentales perseguidos con la realización del mismo.

Capítulo 2. Estado del Arte: Este capítulo tiene como objetivo presentar las distintas tecnologías que constituyen la base para el desarrollo de este proyecto. Se presenta la plataforma para la que se han desarrollado las distintas funcionalidades que componen este proyecto: la red social Twitter, así como el fenómeno de monitorización de usuarios en las redes sociales junto con la aparición de los llamados **bots sociales**.

Capítulo 3. Solución adoptada para la resolución del problema: En este capítulo se representará la arquitectura implementada para el desarrollo del presente proyecto, así como las tecnologías y el entorno de desarrollo empleados para llevarlo a cabo.

Capítulo 4. Análisis de la aplicación TweetyBots: En este capítulo se analizan los parámetros de configuración necesarios para la puesta en marcha del proyecto, así como los distintos casos de usos de la plataforma Web que se ha desarrollar para configurar dichos

parámetros por parte del usuario final.

Capítulo 5. Diseño de la aplicación Web TweetyBots: En este capítulo se realizará una descripción detallada del proceso de diseño de la aplicación Web que utilizará el usuario para crear nuevos bots y configurar su comportamiento.

Capítulo 6. Diseño del comportamiento de los bots: En este capítulo se detallará el proceso de diseño de las funcionalidades de los distintos bots. Igualmente en este capítulo se mostrará el diseño del modelo de datos que hace posible el intercambio de información entre los distintos procedimientos.

Capítulo 7. Pruebas: En este capítulo se incluye el plan de pruebas llevado a cabo para la valoración de los resultados obtenidos.

Capítulo 8. Gestión del proyecto: En este capítulo se muestra la planificación del proyecto, describiendo las tareas a seguir así como el tiempo planificado para cada una de ellas. Se añade también un presupuesto orientativo del coste de la aplicación.

Capítulo 9. Conclusiones y trabajos futuros: En este capítulo, se exponen las reflexiones acerca de la consecución de los objetivos iniciales del proyecto y los resultados obtenidos. Finalmente, se esbozan líneas de trabajo futuras para la posible ampliación del proyecto.

Capítulo 2

Estado del arte

2.1. Introducción

Este capítulo tiene como objetivo presentar las distintas tecnologías que constituyen la base para el desarrollo de este proyecto. Aunque en esta introducción no se incluyan a priori algunos de los términos utilizados, es objetivo de este capítulo que a la finalización del mismo se comprendan en su mayor parte.

En primer lugar, se presenta la plataforma para la que se han desarrollado las distintas funcionalidades que componen este proyecto: la red social **Twitter**. Se englobará dentro de la evolución de las redes sociales para que se entienda mejor cuál es su origen.

A continuación, se hablará del fenómeno de monitorización de redes sociales para la obtención de información útil a partir de conversaciones e interacciones en línea de sus usuarios, haciendo hincapié en las posibilidades que ofrece la obtención de datos mediante el cruce de información social, de la famosa red de microblogging.

Puesto que el objetivo general de este proyecto es la creacción de usuarios robotizados en Twitter de apariencia humana, para analizar su evolución de popularidad de acuerdo a los distintos perfiles de comportamiento observados en Twitter, se analizará el comportamiento de los distintos usuarios en Twitter, agrupando a los usuarios en perfiles, así como categorizando el tipo de publicaciones que se realizan en dicha plataforma.

Para terminar este capítulo, se presentará el principal objeto de estudio de este proyecto, la aparición de programas automatizados que se comunican a través de la plataforma Twitter, conocidos como bots. Para comprender mejor este fenómeno, se presentará la experiencia práctica de contrucción de un bot social, es decir, un bot con apariencia humana que interactúa con el resto de usuarios de la plataforma, y las conclusiones derivadas de este experimento de comunicación social, para llegar a comprender que los bots pueden llegar a jugar un papel fundamental en las comunicaciones de las redes sociales, vista su gran influencia en las mismas.

2.2. Redes sociales

> "Las redes sociales son formas de interacción social, definida por un intercambio dinámico entre personas, grupos e instituciones en contextos de complejidad. Un sistema abierto y en construcción permanente, que involucra a conjuntos que se identifican en las mismas necesidades y problemáticas y que se organizan para potenciar sus recursos." (*Dr. Gustavo Aruguete*) [Aru10]

Una red social es una estructura social en donde hay individuos que se encuentran relacionados entre sí. Las relaciones pueden ser de distinto tipo, como intercambios financieros, amistad, compartición de información, entre otros. Las redes sociales 2.0 son a su vez sitios Web que ofrecen diversos servicios y funcionalidades de comunicación para mantener en contacto a los usuarios que componen la red. Dichas redes se basan en un software especial que integra numerosas funciones individuales: blogs, wikis, foros, chat, mensajería, etc. en una misma interfaz y que proporciona la conectividad entre los diversos usuarios de la red [Eli10].

Figura 2.1: Redes sociales

El fin principal que ha motivado la creación de las llamadas redes sociales, ha sido el de crear comunidades virtuales en las cuales el contenido es aportado y compartido por los integrantes de la comunidad, lo que ha dado lugar al fenómeno de diversos lugares de interacción virtual, en el que millones de personas alrededor del mundo se concentran con diversos intereses en común.

¿Pero cuándo surgió la primera necesidad de establecer contacto con otros a través de Internet? Pues bien, se conoce que en 1971 se envió el primer mensaje de correo electrónico. Se trataba del primer e-mail enviado entre dos ordenadores que estaban uno al lado del otro. Siete años más tarde, en 1978 nacería el BBS (Black Board System) o Sistema de Tablón de Anuncios, un software que utilizarían las redes de ordenadores para conectar a los usuarios de un sistema a través de Internet. De esta manera, tenían la posibilidad de leer noticias, intercambiar información o incluso mensajes con otros usuarios del sistema. Sólo podía acceder al BBS una persona a la vez.

Diez años más tarde, en 1988 se desarrollaría el sistema IRC (Internet Relay Chat) que permitíría a sus usuarios compartir archivos, links y mantenerse en contacto. Pero todavía tendrían que transcurrir algunos años hasta que en 1994 apareciera uno de los primeros sitios web dedicados al social networking. Estamos hablando del servicio Geocities, ahora sumido en el más intenso de los olvidos. Geocities era un servicio de webhosting que permitia a los usuarios crear sus propios sitios Web alojandolos en uno de sus seis "barrios" disponibles.

1997 sería otra fecha clave para los usuarios de Internet. Nació AOL Instant Messenger y se popularizó la mensajería instantánea que permitiría a los usuarios el intercambio de conversaciones en tiempo real. Actualmente sigue persistiendo AOL, aunque en España son más populares servicios como Windows Live Messenger de Microsoft o el GTalk de Google.

Este mismo año nacería SixDegrees, un servicio de red social que permitiría a sus usuarios crear perfiles y establecer relaciones de amistad con otros usuarios. Dicha red estaría activa hasta el año 2001.

En 2003, concretamente, y tras la crisis de la burbuja de las empresas punto com nacería MySpace, un servicio que se codificó en tan solo 10 días y que en el año 2006 llegaría a ser la red social más popular del mundo. En los mismos años surgieron redes sociales tan importantes como Friendster (idea que sirvió para crear luego MySpace), Tribe.net, LinkedIn (una de las primeras redes sociales de negocios), Hi5, Second Life y Del.icio.us, por citar algunos de los ejemplos más conocidos.

Un año más tarde, en 2004, nacería Facebook, la red social que cuenta en nuestros días con el mayor número de usuarios registrados. Se lanzó como un servicio que en un principio tenía que funcionar para los estudiantes de la Universidad de Harvard, y que no cabe duda, triunfó en todo el mundo. Este mismo año, nacería Flickr, un servicio que permite a sus usuarios el intercambio de fotografías.

Al año siguiente, Youtube se convertiría en el sitio Web de alojamiento e intercambios de videos más grande del mundo. Hasta la fecha, de media, se alojan 35 horas de contenido multimedia cada minuto.

Y finalmente, en 2006, se fundó Twitter, la famosa red social de microblogging [1]. Twitter cuenta ahora con alrededor de 200 millones de usuarios que generan más de 65 millones de tweets al día.

A día de hoy, nace la plataforma híbrida Spling, red social de microblogging, que en los cuatro primeros días después de su lanzamiento, el sitio generó alrededor de 1 millón de hits, con 5000 visitantes y 125.000 visitas. Por ello, en la actualidad, se la considera ya como una de las redes de comunicación social de más rápido crecimiento [Nie11] [Hil11].

[1] Microblogging, también conocido como nanoblogging, es un servicio que permite a sus usuarios enviar y publicar mensajes breves (alrededor de 140 caracteres), generalmente sólo de texto. Las opciones para el envío de los mensajes varían desde sitios Web, a través de SMS, mensajería instantánea o aplicaciones ad hoc

En la Figura 2.2 se puede observar la comparación del uso de las principales redes sociales. en estos momentos Facebook ya cuenta con más de 600 millones de usuarios, mientras que Twitter se apunta 190 millones. MySpace sigue siendo la segunda de abordo con 260 millones, aunque el número de seguidores va disminuyendo progresivamente. De momento, se puede considerar que Facebook y Twitter lideran el sector de las redes sociales [Hil11].

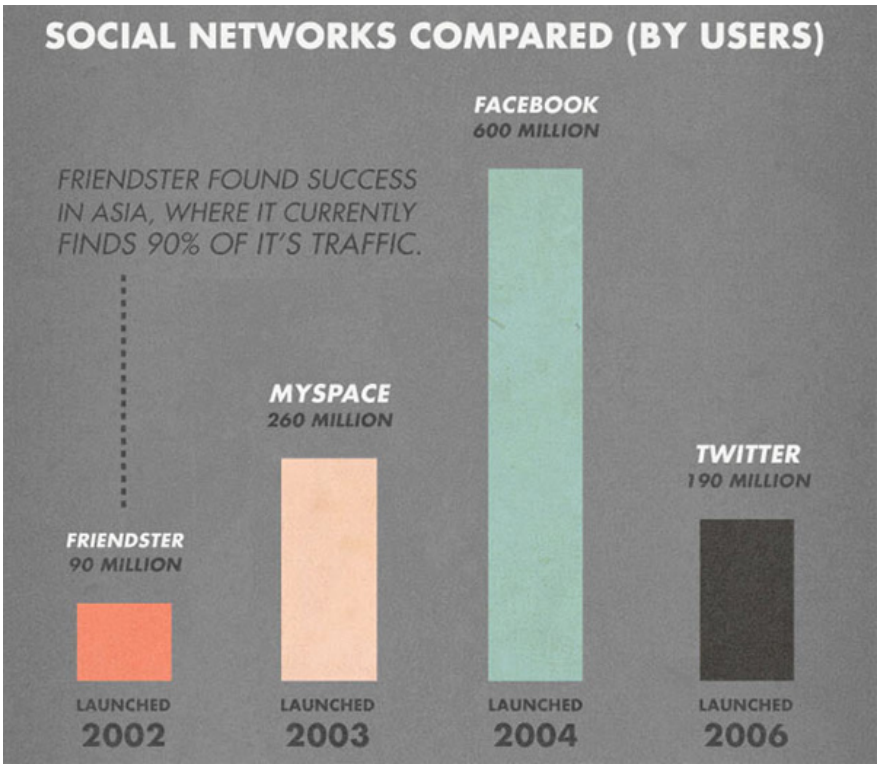


Figura 2.2: Comparacion de las principales redes sociales (Datos Abril 2011)

2.3. Twitter

Dentro del marco de las redes sociales se encuentra Twitter, red social gratuita y servicio de microblogging que permite a sus usuarios enviar "actualizaciones" a través de SMS, mensajería instantánea, correo electrónico, la plataforma Web de Twitter o bien a través de aplicación externas, también conocidas como aplicaciones de escritorio (Ej.: TweetDeck [2]).

Su característica distintiva es su extraordinaria simplicidad. Sus usuarios interactúan a través de publicaciones de mensajes de texto, que se conoce con el nombre de ***Tweet*** (trino, gorjeo) y cuyo tamaño está limitado a 140 caracteres. Puesto que Twitter es un sistema basado en mensajes de texto, es natural que se le compare con otros sistemas igualmente basados en mensajes de texto, como son la mensajería instantánea o bien los servicios de chat. Efectivamente, Twitter tiene una longitud de mensajes similar a los mensajes

[2] `http://www.tweetdeck.com/beta/`

instantáneos o los servicios de chat, pero sin embargo, Twitter carece de "presencia" (cuando los usuarios aparecen conectados/desconectados), ofrece un mayor número de servicios de acceso a su plataforma (Web, SMS, APIs) para leer y enviar actualizaciones, y dispone de contenido persistente.

Twitter fue fundada en el año 2006 por la compañia Obvious Corp, en San Francisco, pero un año más tarde, en abril de 2007, los fundadores originales de Twitter se separaron de dicha compañia y fundaron Twitter, Inc, como compañia independiente, con Jack Dorsey (cofundador de Twitter) como consejero delegado, y que años más tarde se convertiría en todo un fenómeno digital, transformándose en una de las compañias Web con una increíble tasa de crecimiento, puesto que dicha plataforma suma aproximadamente 370.000 usuarios por día.

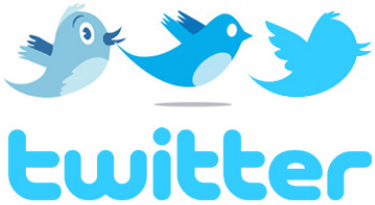


Figura 2.3: Evolución del logo de Twitetr

Pero, ¿a qué se debe su éxito? Hay quien asegura que el éxito de Twitter reside en la limitación de 140 caracteres de sus "actualizaciones" o *tweets*, puesto que se incita al usuario a hacer gala de su máxima expresividad en el mínimo espacio, haciendo de cada uno de ellos algo interesante que merece la pena leer. Otros aseguran que la clave del éxito reside en el caracter asímetrico de las relaciones entre usuarios, puesto que las conversaciones son generalmente de caracter público (se pueden privatizar pero generalmente no se hace), lo que permite a cualquier usuario leer las conversaciones del resto, sin el consentimiento previo del otro [C.G09].

Así mismo, desde sus inicios, uno de los factores de su éxito ha sido la posibilidad de acceder a una API[3] libre que permitió a los desarrolladores explotar su creatividad y llevar la plataforma a unos límites insospechados.

En la actualidad, gran cantidad de industrias usan Twitter como nuevo canal de comunicación. Sin ninguna duda, Twitter ha ayudado a transformar la manera en la cual se obtienen y distribuyen las noticias, ha dado una nueva forma a la manera en la cual se comunican las figuras públicas, desde las estrellas del espectáculo hasta los líderes políticos, y ha desempeñado un importante papel en las protestas populares en Irán, China y Moldavia. También se ha vuelto tan fuerte y omnipresente que ahora compite con Google y Facebook, no sólo por usuarios, también por los dólares que se obtienen a partir de la publicidad [CM10].

[3] `http://dev.twitter.com/doc`

¿Que ofrece Twitter al usuario?

El contenido de los Tweets invita a contestar a la pregunta "¿Qué está pasando?" (What's Happening), reflejo de su interés por ser una herramienta para la divulgación de información en tiempo real. Pero los usuarios de Twitter también responden a otras cuestiones o le dan otros usos [Con09]:

1. Comunicación personal: Era su uso inicial. Mensaje sólo de texto diciendo lo que se está haciendo, leyendo, música que se está escuchando, etc. De hecho la pregunta inicial de "¿Qué estás haciendo?" (What are you doing?) invitaba a los usuarios a contar al mundo las actividades que estaban realizando en un momento dado.

2. Comunicación de grupos de interés: Nace junto con el ***Hashtag***, y se utiliza para comunicación en torno a un tema concreto. Muy usado en eventos que interactúan con Twitter, para investigar o para conocer gente con intereses similares.

3. Difusión de información: Mensajes de recomendación de lectura de un post o artículos mediante una URL. Es un mecanismo muy utilizado para difundir los posts del blog del propio usuario. No se considera autobombo. Existen aplicaciones que traducen URLs largas en otras más cortas, conocidas como acortadores URL (Ej.: tinyurl.com, bit.ly) que permiten aprovechar al máximo los 140 caracteres del mensaje.

4. Compartir información gráfica y visual: Compartir fotos y videos. Twitter no permite subir imágenes (de momento) y video, pero es posible publicarlas por medio aplicaciones externas como son Twitpic.com o yfrog.com, o bien compartir enlaces a videos de plataformas como Youtube, que también implementa su propio sistema acortador de URLS, adaptándose de este modo a compartir información en este tipo de plataformas.

2.3.1. Monitorización de las redes sociales

Debido a la cantidad de usuarios que se involucra a diario en conversaciones e interacciones en línea a través de las redes sociales, el mercado de herramientas de redes sociales se ha concentrado en soluciones que permitan evaluar la situación general, detectar tendencias y administrar la comunicación en la porción muchos-a-muchos de la comunicación.

De este fenónemo surge la idea de la monitorización de dichas redes y la creacción de nuevas herramientas que permitan la extracción de información útil para su posterior procesado y análisis para la realización de distintos tipos de informes sobre distintos temas, tales como: ranking de personas influyentes, mensajes destacados, alcance geográfico, palabras destacadas, actividad generada, perfil de los usuarios, etc.

¿Merece la pena investigar sobre el contenido de Twitter?

Si uno se detiene a pensar a analizar la cantidad de información que se podría obtener de Twitter se sorprendería de que de algo tan simple sea posible cruzar tanta información de interacción social [Con09].

Debido a la cantidad de información útil que se puede obtener de Twitter, y gracias al API que pone dicha plataforma a disposición de los desarrolladores de aplicaciones (Twitter ha publicado un conjunto de funciones que soportan la recogida de información de los usuarios), han surgido diversas herramientas de monitorización de usuarios, hashtags, tendencias, marcas, etc., de dicha red social, que mejoran la experiencia del usuario y permiten explotar al máximo las posibilidades ofrecidas por la plataforma.

Por ejemplo, son famosas las aplicaciones destinadas a monitorizar las tendencias en Twitter, esto es, un tema sobre el que se está twitteando mucho, es decir, hay mucha gente en Twitter hablando sobre ello.

La utilidad de una funcionalidad así se podría ilustrar con ejemplos como:

- Seguir los acontecimientos posteriores a un desastre natural mediante los tweets de personas próximas al area donde tuvo lugar.
- Seguir eventos públicos o deportivos mediante los tweets de personas que estan presentes en el lugar o lo estan viendo a través de algún medio de comunicación.
- Monitorizar qué se dice sobre una determinada compañía en Internet en favor del éxito empresarial de una marca o producto.

Además, si se combina dicha funcionalidad con la posibilidad de restringir las búsquedas a ámbitos geográficos concretos, se puede saber qué se está diciendo en un sitio concreto y en un momento dado.

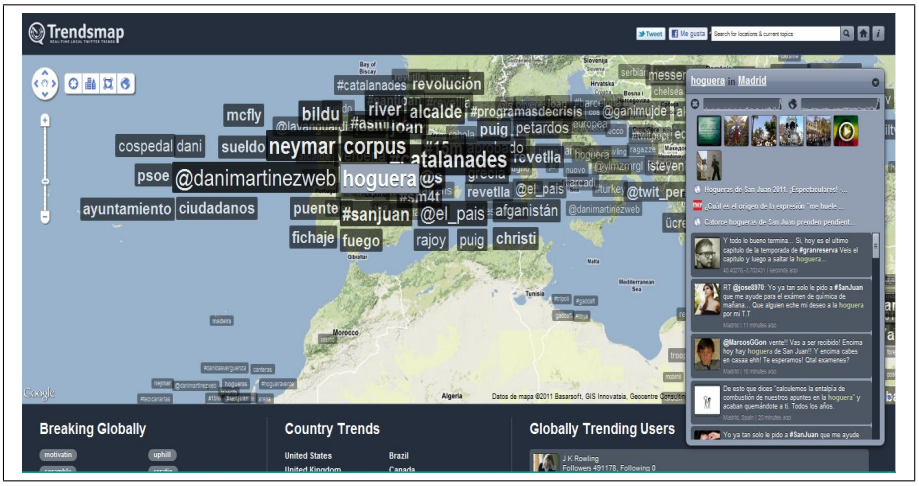
Desde que apareció esta funcionalidad ya hace algún tiempo, han aparecido un gran número de páginas y servicios web que, mediante la API de Twitter, ofrece a los usuarios la posibilidad de explorar estas tendencias de formas tan creativa como cuantitativamente.

Como ejemplos de aplicaciones orientadas a la monitorización de tendencias en Twitter, se pueden encontrar Trendsmap [4], Revisit [5] y Monitter [6], entre otras. [Dur10]

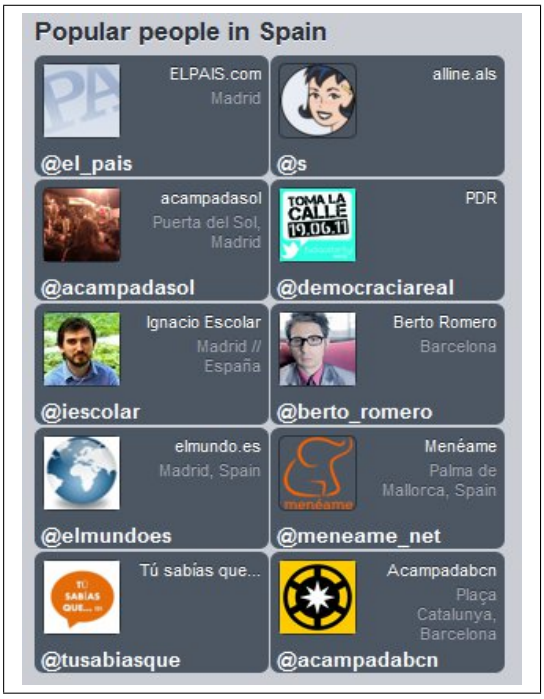
Trendsmap es una de las aplicaciones Web más atractivas visualmente a la hora de buscar e identificar tendencias de conversación en Twitter. Dicha aplicación permite al usuario obtener los ***Trending topics*** de Twitter en función de una localización dada, así como los usuarios más destacados, los links más interesantes, las imágenes y los videos favoritos de dicha localización.

[4] `http://trendsmap.com/`
[5] `http://moritz.stefaner.eu/projects/revisit/`
[6] `http://www.monitter.com/`

En la Figura 2.4 se muestra un ejemplo de los trending topics localizados en España, así como las personas más populares en dicho país, en en un momento dado, proporcionados por la aplicación TrendsMap.



(a) Trending Topics en España (Junio 2011)



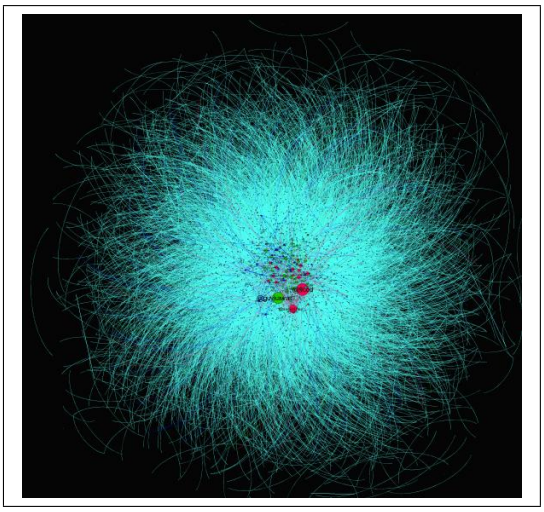
(b) Personas más populares en España (Junio 2011)

Figura 2.4: Capturas de pantalla de la herramienta TrendsMap

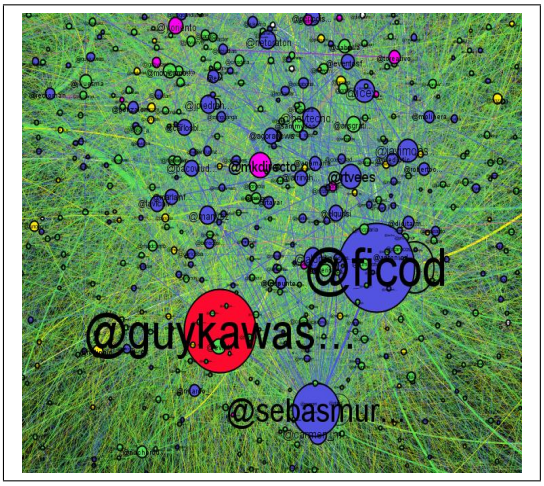
Del mismo modo, el interés por la monitorización de usuarios y conversaciones en Twitter, puede verse también en divulgaciones cientificas que estudian las relaciones entre usuarios, experimentos sociales, predicciones, cálculo de influencias de usuarios, estadísticas, entre otros.

Como ejemplo de lo que se puede llegar a conseguir mediante la monitorización de usuarios y conversaciones en Twitter, se encuentra el experimento llevado a cabo por M.Luz congosto, investigadora de la Universidad Carlos III de Madrid, en el que se analizan las conversaciones relacionadas con uno de los muchos eventos que se organizan a través de la red social de Twitter, para realizar diferentes comparaciones entre ellos.

En la Figura 2.5 se muestra uno de los grafos obtenidos durante el experimento que muestra la red que se forma con las menciones de los usuarios que participaron en dicho evento [Con10a].



(a) Red de menciones entre usuarios (I)



(b) Red de menciones entre usuarios (II)

Figura 2.5: Red de menciones entre usuarios del evento FICOD 2010

2.4. Comportamiento de los usuarios en Twitter

Como el objetivo final de este proyecto es el poder crear un bot con apariencia humana para poder analizar posteriormente el valor que se le da en Twitter al comportamiento de los usuarios, es necesario establecer cuáles son las acciones que un usuario realiza normalmente en Twitter.

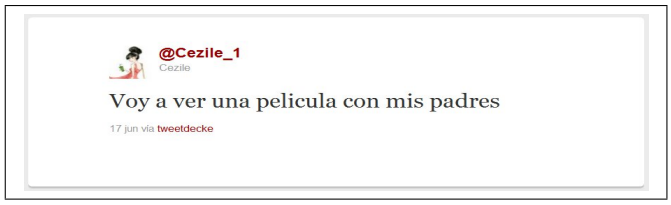
2.4.1. Acciones que puede realizar un usuario de Twitter

Las acciones que un usuario puede realizar en Twitter se pueden dividir en dos grupos: Realizar publicaciones y establecer relaciones.

REALIZAR PUBLICACIONES

A continuación se enumeran los tipos de publicaciones que un usuario puede realizar en Twitter.

- Actualización de estado: Esto es cuando un usuario publica un ***Tweet*** para que sus seguidores de Twitter puedan leerlo a través de su ***Timeline*** público. En la Figura 2.6 se pueden observar dos ejemplos de estados en Twitter.



(a) Ejemplo actualización estado en Twitter (I)

(b) Ejemplo actualización estado en Twitter (II)

Figura 2.6: Ejemplos actualización estado en Twitter

- Retransmitir un estado: Esto es cuando un usuario realiza un ***ReTweet (RT)*** al estado de otro usuario para que sus seguidores de Twitter puedan leerlo a través de su timeline público.

  Tal y como se observa en la Figura 2.7 hay 2 formas de realizar un RT a un usuario en Twitter:

- La manera tradicional: Se antepone el texto "RT:" al alias en Twitter del usuario seguido del texto del estado que se quiere retransmitir.
- El nuevo sistema de RT de Twitter: Hay un botón a tal efecto en la aplicación que permite retransmitir el estado de un usuario sin necesidad de modificarlo a mano. De esta forma se pierde el caracter social del "RT" tradicional, por lo que esta opción es menos usada por los usuarios de Twitter, pero se recurre a ella cuando al hacer un RT tradicional de un estado se supera la limitación de 140 caracteres por tweet impuesta en Twitter.

Figura 2.7: Ejemplos RTs en Twitter

- Responder a un estado previo de un usuario en Twitter (***Reply***). Para ello, se antepone el alias en Twitter del usuario propietario del estado al que se quiere responder, seguido del texto de la respuesta. En la Figura 2.8 se muestra un ejemplo de Reply en Twitter.

Figura 2.8: Ejemplo de respuesta a estado en Twitter

- Enviar un mensaje privado (***Direct Message (DM)***) a un usuario: Para poder enviar un mensaje privado para que únicamente lo pueda leer la persona destinataria del mensaje. Estos mensajes se pueden visualizar a través de la página personal de un usuario de Twitter (en la opción "Mensajes" anclada en la barra de herramientas de Twitter) y también se reciben por medio de una notificación a la dirección de correo electrónico indicada durante el registro en Twitter del usuario destinatario.

ESTABLECER RELACIONES

Las relaciones que se pueden establecer entre los usuarios de Twitter son asimétricas, esto es, que para que dos usuarios puedan compartir información no tiene que haber un consentimiento mutuo entre ambos, que es lo normal en las plataformas de redes sociales.

En Twitter, cada usuario tiene un conjunto de usuarios favoritos, denominados ***Following***, a los que puede ver sus tweets y a su vez posee un conjunto de usuarios de los que es favorito, llamados ***Followers***, que pueden leer lo que él escribe. Cuando dos usuarios de Twitter se corresponden, es decir, ambos siguen los tweets del otro se les denomina ***Friends***. Aunque en la plataforma únicamente se muestran los Following y los Followers de un usuario, se pueden deducir los Friends de ambos conjuntos [Con09].

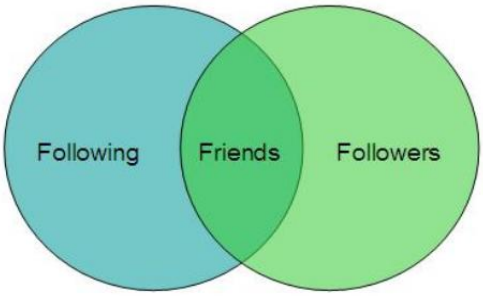


Figura 2.9: Diagrama de las relaciones entre usuarios en Twitter

2.4.2. Fuentes de información

Tal y como se ha visto en el Apartado 2.4.1, los usuarios en Twitter pueden realizar publicaciones de distinto tipo, y para ello se basan en distintas fuentes de información. A continuación se enumeran las principales:

- Medios de comunicación Web: A través de Twitter los usuarios publican las noticias que consideran más relevantes de distintos tipos de periódicos o revistas digitales.
- Lista de blogs: A través de Twitter los usuarios publican curiosidades publicadas en distintos blogs.
- Hashtags: Son palabras claves que permiten etiquetar un tweet para facilitar su búsqueda al resto de usuarios de la plataforma. Se forman con el caracter "#" seguido de la etiqueta que se quiere asignar a un determinado tweet. (Ej.: #rmadrid). Gracias a estas palabras clave los usuarios pueden realizar búsquedas explícitas en Twitter y retransmitir a través de su timeline aquella información que consideren relevante.
- Gurús de Twitter: Son personas que tienen un alto número de seguidores en Twitter y que siguen sólo a unos pocos usuarios. Suelen ser personas influyentes en Twitter que suelen publicar noticias con alto contenido informativo y de gran interés para sus followers. Por ello se les considera grandes fuentes de información.

- Herramientas de compartición de contenido multimedia: A través de estas herramientas los distintos usuarios de Twitter transmiten contenido multimedia a través de la plataforma: videos, fotos, etc. Entre las herramientas más usadas para compartir contenido multimedia en Twitter se encuentran: Youtube[7] para la compartición de videos o Twitpic[8] (videos y fotos), etc.
- Herramientas para compartir información de localización: A través de estas herramientas los distintos usuarios de Twitter transmiten información de localización a sus seguidores a través de la plataforma. Entre las herramientas más utilizadas para compartir información de localización en Twitter se encuentran: FourSquare[9], Endomondo[10], etc.
- Mensajes de caracter personal: opiniones, quejas, pensamientos aleatorios, anécdotas, etc.

2.4.3. Tipos de contenidos

Une vez vistas las acciones que un usuario puede realizar en Twitter, conviene categorizar los tipos de contenidos públicos que un usuario puede publicar a través de la plataforma.

A partir de un estudio cuantitativo realizado por investigadores de la Universidad Rutgers, de los mensajes publicados por 350 usuarios de Twitter, se llegarón a clasificar los mensajes publicados en Twitter en 9 categorías.[MN10] (véase Cuadro 2.1):

- **Compartición de información** (*Information Sharing* [IS]): En este tipo de mensajes el usuario transmite/ retransmite información que considera de interés para que sus followers puedan leer a través de su timeline público. Este tipo de mensajes se caracterizan por:
  - Transmitir links a noticias en medios de comunicación o revistas electrónicos.
  - Transmitir links a articulos de blogs.
  - Retransmitir estados que contengan hashtags favoritos del usuario.
  - Retransmitir estados de gurús favoritos del usuario.
  - Retransmitir estados de medios de comunicación favoritos del usuario presentes en la plataforma (Ej.: @CNN, @elmundo, @elpais, ...).
  - Retransmitir estados de blogs favoritos del usuario presentes en la plataforma (Ej.: @mashable, @engadget, ...)
- **Autopromoción** (*Self Promotion* [SP]): En este tipo de mensajes el usario promociona páginas personales, blogs, productos, etc., para promover que sus followers realicen visitas a sus páginas personales, compren sus productos, etc.; e incluso suele animar a nuevos usuarios a que le sigan en Twitter.

[7] `http://www.youtube.es`
[8] `http://twitpic.com/`
[9] `https://foursquare.com`
[10] `http://www.endomondo.com`

- **Opiniones/Quejas** (*Opinions/Complaints* [OC]) y **Declaraciones y pensamientos aleatorios** (*Statements and Random Thoughts* [RT]): En este tipo de mensajes el usuario muestra su conformidad/disconformidad con temas actuales, vierte sus opiniones, o bien realiza cualquier tipo de declaración o pensamiento aleatorio.
- **Que estoy haciendo ahora** (*Me now* [ME]): En este tipo de mensajes el usuario mantiene informados a sus seguidores de Twitter de las actividades que se encuentra realizando en un momento determinado.
- **Preguntas a followers** (*Question to followers* [QF]): En este tipo de mensajes el usuario lanza preguntas abiertas para que sus seguidores en Twitter participen vertiendo sus opiniones sobre un determinado tema.
- **Presencia** (*Presence Maintenance* [PM]): En este tipo de mensajes, el usuario informa a sus seguidores de que se encuentra en un determinado momento utilizando la plataforma o bien que ha dejado de usarla.
- **Anécdotas propias** (*Anecdote*(*me*) [AM]) y **Anécdotas de otros** (*Anecdote* (*others*) [AO]): En este tipo de mensajes los usuarios cuentan anécdotas de sí mismos o de terceras personas que quieren transmitir a sus seguidores en Twitter.

Cuadro 2.1: Categorización de mensajes en Twitter

Tipo de mensaje	**Ejemplo(s)**
Compartición de información (*Information Sharing* [IS])	"Los chistes fáciles de Matías Prats son sencillamente espectaculares http://bit.ly/iXQvHn"
Autopromoción (*Self Promotion* [SP])	"Visita mi blog: http://g33kca.wordpress.com/"
Opiniones/Quejas (*Opinions/Complaints* [OC])	"Los problemas del #Atleti son dos y están en el palco"
Declaraciones y pensamientos aleatorios (*Statements and Random Thoughts* [RT])	"Hoy hace un día perfecto para salir a correr"
Que estoy haciendo ahora (*Me now* [ME])	"Estoy haciendo la compra con mis padres"
Preguntas a los followers (*Question to followers* [QF])	"Estais de acuerdo con el nuevo fichaje del #Atleti?"
Presencia (*Presence Maintenance* [PM])	"Buenos dias a tod@s"
Anécdotas propias (*Anecdote (me)* [AM])	"Esta mañana me he quedado encerrado en el ascensor"
Anécdotas de otros (*Anecdote (others)* [AO])	"A mi padre le han tocado 3 números de la bonoloto!!"

Igualmente de dicho estudio se extrae la conclusión de que la mayor parte de los mensajes intercambiados entre los usuarios de Twitter corresponde a la categoría "Qué estoy haciendo ahora", así como a mensajes de tipo "Compartición de información", "Opiniones/Quejas" y "Declaraciones y pensamientos aleatorios".

En la Figura 2.10 se muestra la frecuencia por categoría de los mensajes intercambiados por los distintos usuarios en Twitter.

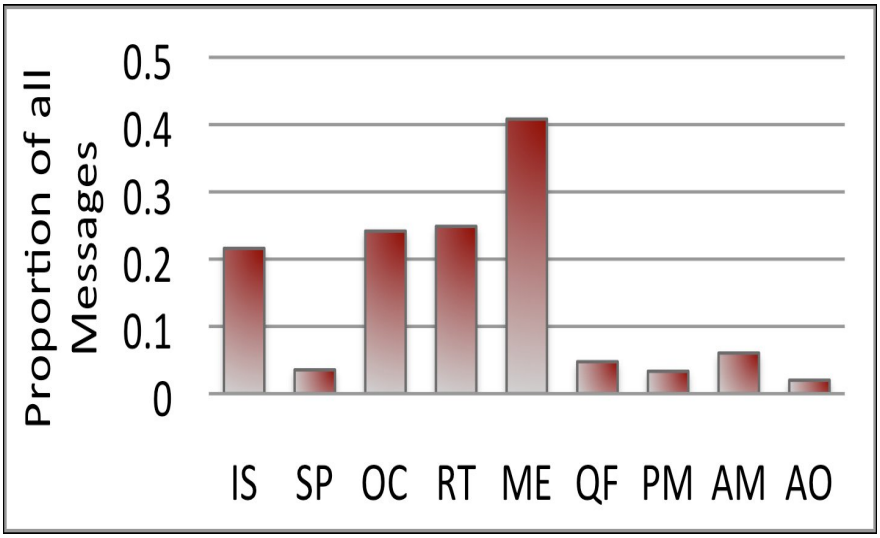


Figura 2.10: Frecuencia de por categorías de mensajes en Twitter

2.4.4. Tipos de Usuarios

Dependiendo de la proporción entre Followings/Followers de un usuario, se podrían agrupar los distintos tipos de usuarios de Twitter en tres perfiles [Con09]:

- Generadores de opinión o locutores: Cuando el número de Followers es muy superior al de Followings. Generalmente hablan más que escuchan.Son normalmente cuentas de celebridades o famosas organizaciones. Este tipo de usuarios se caracteriza por generar contenidos interesantes y atraer a numerosos suscriptores (Followers). Este tipo de usuarios reciben una alta proporción de ***Menciones (Mentions)*** de otros usuarios.
- Fan o spammers: Cuando el número de de Followings es muy superior al número de Followers. Escuchan más que hablan. Este tipo de perfil está generalmente asociado a bots de spam en Twitter.
- Friendly o conocidos: cuando la proporción entre Followings y Followers se encuentra compensada. Es el perfil más común.

2.5. Bots en Twitter

La popularidad y la estructura abierta de Twitter han atraido a un gran número de programas automatizados que se sirven del API de twitter y de las herramientas disponibles en el mercado para realizar publicaciones y seguir a usuarios en Twitter de manera automática. Es lo que se conoce como bots de Twitter. Algunos bots generan una gran cantidad de Tweets con contenido benigno, como publicación de noticias o actualizaciones de fuentes Rss, mientras que otros bots malintencionados propagan mensajes de spam y contenidos maliciosos. Este último tipo de bots suelen añadir de manera aleatoria gran cantidad de usuarios a sus listas de Followings, con la esperanza de que alguno de ellos les corresponda, de este modo los mensajes de tipo spam publicados por el bot aparecería en el Timeline de los usuarios que le siguen.

Twitter no inspecciona estrictamente las aplicaciones de automatización, únicamente requiere durante el proceso de registro, que el usuario sea capaz de reconocer una imagen CAPTCHA[11]. Una vez registrado el usuario en la aplicación, se puede diseñar cualquier tipo de bot que imite las tareas que realiza un humano en Twitter, únicamente realizando llamadas al API de Twitter.

Pero, ¿qué comportamiento diferencia a un usuario humano de un bot?

En general, un usuario de Twitter se puede clasificar como humano si muestra alguna evidencia de originalidad o inteligencia humana en sus publicaciones. En particular, un usuario humano responde con frecuencia a la pregunta "¿Qué estás haciendo?" y utiliza la herramienta para mostrarse a sí mismo e interactuar con sus amigos, mientras que un bot suele realizar constantemente *ReTweets* a los estados de otros usuarios o bien publicar actualizaciones de fuentes Rss. Algunos bots incluso publican con excesiva automatización y abundancia. Otra característica de los bots es la repetición de publicaciones produciéndose de este modo la duplicación de sus estados. Incluso se ha visto el caso de actualizaciones de estados de bots con links que no correspondían con el contenido del *Tweet*. Finalmente, lo que más caracteriza el comportamiento de los bots es su manera agresiva de añadir *Followings* a sus cuentas, con el fin de llamar la atención de los usuarios humanos, los bots realizan *Followings* de manera masiva y del mismo modo realizan *Unfollowings* (dejar de seguir a un usuario en Twitter) al cabo de un tiempo.

[11] CAPTCHA es el acrónimo de Completely Automated Public Turing test to tell Computers and Humans Apart (Prueba de Turing pública y automática para diferenciar máquinas y humanos). Se trata de una prueba desafío-respuesta utilizada en computación para determinar cuándo el usuario es o no humano. La típica prueba consiste en que el usuario introduzca un conjunto de caracteres que se muestran en una imagen distorsionada que aparece en pantalla. Se supone que una máquina no es capaz de comprender e introducir la secuencia de forma correcta por lo que solamente el humano podría hacerlo

¿Los bots tienen mayor número Followings que de Followers?

Dependiendo del porcentaje *Followers/Followings*, se podría categorizar a los usuarios de Twitter en 3 grupos:

- Grupo I: En los que el número de *Followers* es claramente mayor al número de *Followings*
- Grupo II: En el que la situación es la contraria. El número de *Followings* es claramente mayor al número de *Followers*
- Grupo III: Donde la situación se encuentra más o menos compensada. El número de *Followings* es similar al número de *Followers*

En base a esta categorización y observando distintos tipos de perfiles de usuarios en Twitter, se puede deducir que las relaciones humanas en Twitter corresponden al tercer grupo, es decir, las relaciones entre humanos en las redes sociales son normalmente recíprocas. Aunque también pueden encontrarse individuos que pertenecen al primer grupo, en los que el número de *Followers* es claramente mayor al número de *Followings*, pero suelen ser cuentas asociadas a celebridades y famosas organizaciones.

Sin embargo los bots en Twitter suelen pertenecer con mayor frecuencia al segundo grupo, en el que el número de *Followers* es claramente mayor que el número de *Followings*, puesto que la estrategia general empleada por los bots es la de seguir a muchos usuarios en Twitter y sin embargo sólo unos pocos les corresponden. Por esta razón Twitter impone un límite en el porcentaje *Followers/Followings* para suprimir este tipo de comportamientos en Twitter [dadT]. Por este motivo, algunos bots avanzados dejan de seguir a los usuarios que no les corresponden en un cierto tiempo.

¿Hay alguna característica temporal en los usuarios de Twitter que diferencie a un bot de un humano?

Igualmente la frecuencia de publicación y los horarios pueden diferenciar el comportamiento de un bot. Mientras que los humanos suelen ser más activos durante los días laborables (L-V), y menos activos durante el fin de semana (S-D), los bots suelen tener el mismo nivel de actividad durante todos los días de la semana. Sin embargo, algunos bots avanzados únicamente publican Tweets en un intervalo de tiempo dado.

Otra característica asociada a la automatización es el comportamiento regular y periódico a la hora de realizar publicaciones, y por ello, este comportamiento regular está asociado a programas automáticos y bots, puesto que los usuarios humanos realizan publicaciones en Twitter de manera irregular, sin seguir ningún tipo de patrón.

¿A través de que plataformas se suelen realizar las publicaciones en Twitter?

Twitter soporta diferentes tipos de medios desde los que realizar publicaciones en su plataforma. Por esta razón se muestra el nombre del dispositivo origen desde el que se

publica cada Tweet en dicha plataforma.

Figura 2.11: Dispositivo origen de publicación

Dichos medios se podrían clasificar en 4 categorías:

1. Plataforma Web: A través de la página Web Twitter.com, en la que el usuario debe introducir manualmente su login y password para realizar publicaciones.

2. Dispositivos móviles: Son programas especiales que se ejecutan desde ciertos dispositivos móviles y que permiten realizar actualizaciones en la plataforma.

3. Aplicaciones de terceros registradas: Dichas aplicaciones utilizan el API de Twitter para realizar publicaciones y deben de encontrarse registradas en dicha plataforma. Dentro de este grupo se encuentran las herramientas de apoyo en Twitter (Twitpic, bit.ly, etc.), extensiones del navegador (Tweetbar), aplicaciones de escritorio (TweetDeck), etc.

4. Aplicaciones de terceros no registradas o no certificadas por Twitter.

Una vez realizada la categorización de diferentes dispositivos que permiten realizar publicaciones, se observa que la mayoría de los usuarios humanos realizan sus publicaciones a través de la plataforma Web de Twitter o bien a través de dispositivos móviles o aplicaciones de escritorio, puesto que la utilización de dichas herramientas requieren la participación de un usuario humano. Sin embargo la mayoría de las herramientas utilizadas por los bots son autómaticas y no requieren de ningún tipo de supervisión humana, por lo que este tipo de usarios suelen realizar publicaciones a través de aplicaciones de terceros no certificadas por Twitter.

¿Los bots suelen incluir mayor número de links externos en sus publicaciones en Twitter?

Efectivamente, la frecuencia con la que un bot incluye enlaces de URLs en sus publicaciones es mayor, e incluso algunos Tweets de este grupo de usuarios suelen incluir más de un enlace. La frecuencia con la que un humano publica Tweets con enlaces es mucho menor, puesto que los humanos suelen realizar publicaciones respondiendo a la pregunta “Qué está pasando” o “Qué estás haciendo?” propuesta por Twitter a sus usuarios [CGWJ10].

2.5.1. Experiencia práctica de construcción de un bot social

A continuación se mostrará la experiencia práctica llevada a cabo por el desarrollador Web Konstantin Kovshenin, para obtener una idea general de lo que hay implementado hasta el momento en el campo de la creacción de bots con apariencia humana en Twitter, cada vez más frecuentes debido a la creciente popularidad y naturaleza abierta de la red de microblogging.

Se ha elegido este ejemplo, porque es el que mejor ilustra la base del objetivo de este proyecto. De hecho partiendo de la idea original de su desarrollador, se tratará de mejorar el modelo propuesto para ampliar su funcionalidad y darle una apariencia más humana.

A continuación se detalla el experimento llevado a cabo por Konstantin Kovshenin.

TwiBots

TwiBots es un proyecto que comenzó en 2009. Dicho proyecto consistía en el desarrollo de un programa autómatico basado en el API de Twitter que realizaría publicaciones en la plataforma todos los días de la semana durante 24 horas. pero dicho programa evolucionó en una simulación de usuario de Twitter que recogía información sobre determinados temas a través del API de búsquedas por palabras clave proporcionado por Twitter, así como ciertos contenidos de fuentes Rss, utilizando palabras claves y expresiones regulares. En la última configuración conocida del proyecto, el bot era capaz de:

- Publicar Tweets durante todo el día.
- Publicar Tweets con información obtenida de fuentes RSS, con funcionalidad añadida de posible adición de hashtags.
- Realizar ReTweets por medio del API Search de Twitter.
- Construir ***Listas*** de conversación.
- Enviar sentencias aleatorias, de un conjunto de frases prefijadas, a los usuarios que le realizan menciones.
- Agradecer los ReTweets realizados por otros usuarios al bot.
- Seguir a los usuarios que realizan ReTweets al bot.

Con dicha configuración, el resultado obtenido por el comportamiento del bot en Twitter al cabo de 6 meses, es realmente asombroso. El bot había alcanzado la asombrosa cantidad de más de 4500 *Followers* contando con poco más de 200 *Followings* y había sido incluido en más de 250 Listas. Durante los 6 meses el bot envió alrededor de 55000 Tweets y ReTweets basados en palabras clave (diseño, diseño Web, Wordpress y JQuery) y construyó 4 Listas basadas en las mismas palabras clave con 500 miembros en cada una.

Igualmente, durante la duración del experimento, los *Followers* del bot interactuaron con él. Los usuarios que seguían al bot hicieron *Reply* a sus mensajes, le agradecieron sus ReTweets, le desearon un buen día e incluso le animaron a tomar un café, mientras que el bot la mayoría de las veces no respondía a las *Menciones*.

A continuación, en la Figura 2.12 se muestra un gráfico de la evolución de la cuenta del bot realizado por Kovshenin durante los últimos 3 meses de recogida de datos del experimento, comparando su evolución con la de la cuenta personal de su autor en Twitter, quien en poco más de un año había conseguido reunir la cifra apróximada de 1700 *Followers* [Kov10].

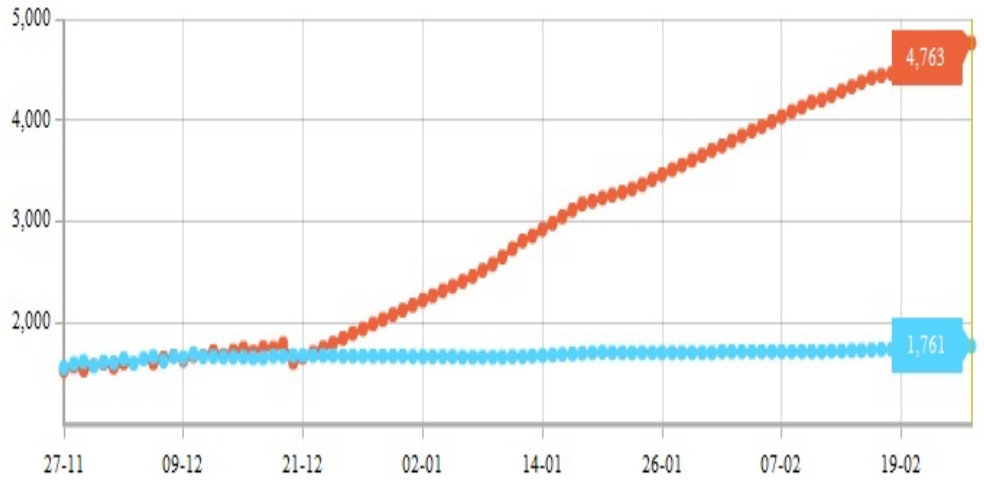


Figura 2.12: Gráfica comparativa Followers conseguidos por el bot

2.5.2. Competiciones de bots sociales en Twitter

Debido a la, cada vez mayor, presencia de bots sociales en Twitter, se ha organizado incluso un evento competitivo, llamado SocialBots2011, en el que se invita a sus participantes a la construcción de bots sociales que se comuniquen en distintas redes sociales, entre ellas Twitter, con los distintos usuarios durante 2 semanas, para realizar un estudio social y averiguar así quién se comunica mejor: los seres humanos o los bots [PRO11].

Para dicha competición, tres grupos anónimos desarrollaron aplicaciones automatizadas con el objetivo de simular el comportamiento de usuarios humanos en Twitter, y que efectivamente logró engañar a un grupo de usuarios a través de Twitter haciéndoles pensar que se trataban de personas reales. Durante el período establecido de dos semanas que duraba la competición, los tres robots sociales, se integraron en un grupo de personas y consiguieron cerca de 250 *Followers* cada uno, pareciendo usuarios humanos mediante el envio de mensajes a otros usuarios y recibiendo respuestas frecuentemente.

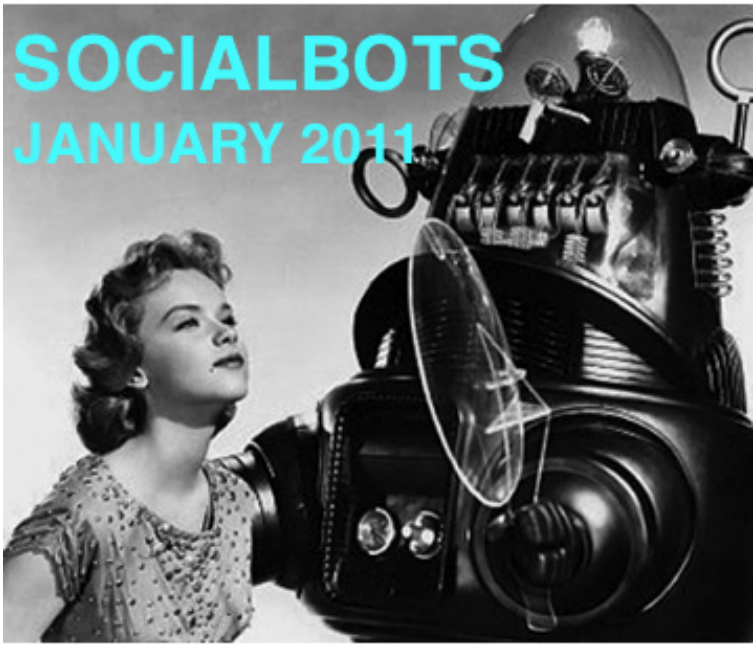


Figura 2.13: Cartel de la competición SocialBots 2011

Para ello, los bots buscaban mensajes de usuarios que en torno a unos mismos intereses y creaban respuestas estándar acordes a lo que estos usuarios habían publicado. Estas interacciones eran suficientemente reales para atraer la atención de un grupo de personas dentro de la comunidad quienes empezaron a seguirlos y responder a sus mensajes.

Lo que demuestra claramente que los bots sociales pueden manipular comunidades de redes sociales para, obviamente, hacer el bien y el mal, apoyando o estando en contra en diversas causas.

Sin duda la inteligencia artificial en este campo todavía tiene mucho que demostrar y sin duda las técnicas para conseguir que los bots sociales se integren en redes sociales e interactúen con sus usuarios mejorarán notablemente con el paso del tiempo [Mar11].

Capítulo 3

Solución adoptada para la resolución del problema

3.1. Introducción

En este capítulo se representará la arquitectura implementada para el desarrollo del presente proyecto, así como las tecnologías y el entorno de desarrollo empleados para llevarlo a cabo.

Para realizar el programa automatizado, es decir, el bot, y que se comporte como un usuario humano en la red social de Twitter, será necesario introducirle ciertos parámetros de configuración, para establecer unas determinadas pautas de comportamiento, haciendo de este modo que el comportamiento de cada bot sea único, atendiendo a diversos criterios de configuración. Para ello, es necesario desarrollar una interfaz que se comunique con el cliente (usuario que quiere crear un nuevo bot), para que configure dichas características.

Por ello, dividiremos el planteamiento de la plataforma en dos partes diferenciadas: por un lado, se encontrará la solución adoptada para desarrollar la interfaz que se comunicará con el cliente para la entrada de datos de configuración de los distintos bots, y por otro lado se encontrará la solución adoptada para desarrollar las distintas funcionalidades que realizarán los bots en la red de microblogging.

3.2. Arquitectura de la aplicación Web

La solución adoptada como interfaz para la comunicación con los distintos usuarios interesados en la creación de bots sociales en Twitter, ha sido la de crear una plataforma Web. De este modo la plataforma es fácilmente accesible a cualquier usuario a través de un navegador, sin necesidad de descargar e instalar ningún tipo de Software en el equipo cliente.

De este modo, la arquitectura de la aplicación Web estará formada principalmente por tres capas. La capa de los clientes, la capa del servidor de aplicaciones y la capa de la base de datos.

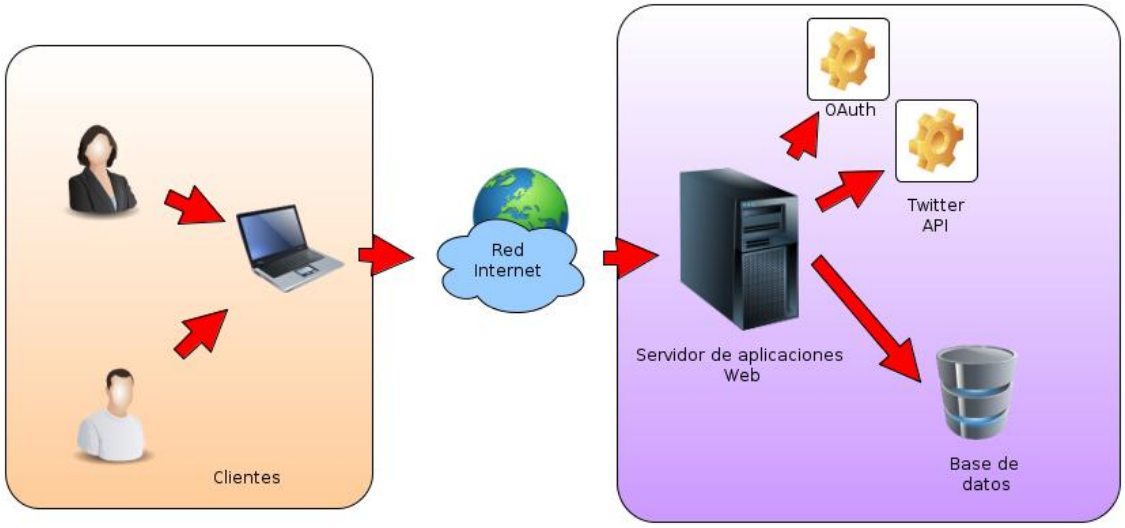


Figura 3.1: Arquitectura plataforma Web

En la capa de los clientes se sitúan todos aquellos usuarios interesados en la creación de bots sociales a través de la herramienta. Los puestos utilizados por los clientes representan un cliente ligero puesto que toda la lógica de negocio residirá en el servidor Web de aplicaciones. Los puestos de los clientes únicamente deben contar con el Software necesario para realizar invocaciones a las aplicaciones residentes en el servidor Web y visualizar su resultado. Esta tarea se realizará de manera sencilla para el cliente a través de un navegador Web, sirviéndose para la comunicación con el servidor del protocolo HTTP.

La capa del servidor Web alberga toda la lógica de negocio y presentación de la aplicación. El servidor se encargará de escuchar las peticiones de los clientes y de enviar las respuestas correspondientes en función del tratamiento particular de cada tipo de petición contemplada. Igualmente, el servidor de aplicaciones incluye la funcionalidad necesaria para realizar peticiones y recibir respuestas, a un servidor donde se aloja la Base de Datos, así como la funcionalidad necesaria para comunicarse con el API de Twitter, para actualizar/recibir datos de la plataforma Twitter, así como para comunicarse con OAuth, protocolo necesario para obtener las claves del usuario en Twitter asociado al bot, y que permitirá que la aplicación de TweetyBots pueda comunicarse con Twitter a través de su API en representación de dicho usuario.

La capa de la Base de Datos, se encarga del almacenamiento persistente de la información necesaria para el correcto funcionamiento de la aplicación.

La comunicación entre la capa de los clientes y el servidor de aplicaciones Web, se realizará a través de la red Internet, al igual que la comunicaciones llevadas a cabo entre el servidor Web y los programas externos (API de Twitter y OAuth). Por otro lado, la comunicación entre la capa del servidor Web de aplicaciones y el servidor de Base de Datos, se realizarán a través de la red interna de la Universidad, puesto que ambos servidores se encuentran instalados en diferentes máquinas de la Universidad.

3.3. Arquitectura para la gestión de las funcionalidades de los bots

La solución adoptada para llevar a cabo la gestión de las distintas funcionalidades de los bots, esto es, las acciones que un bot podrá realizar en Twitter, es la de desarrollar un conjunto de procedimientos/tareas que se albergarán en un servidor Web de aplicaciones y que serán gestionado/as por un mecanismo que lance periódicamente la ejecución de cada procedimiento en dicho servidor.

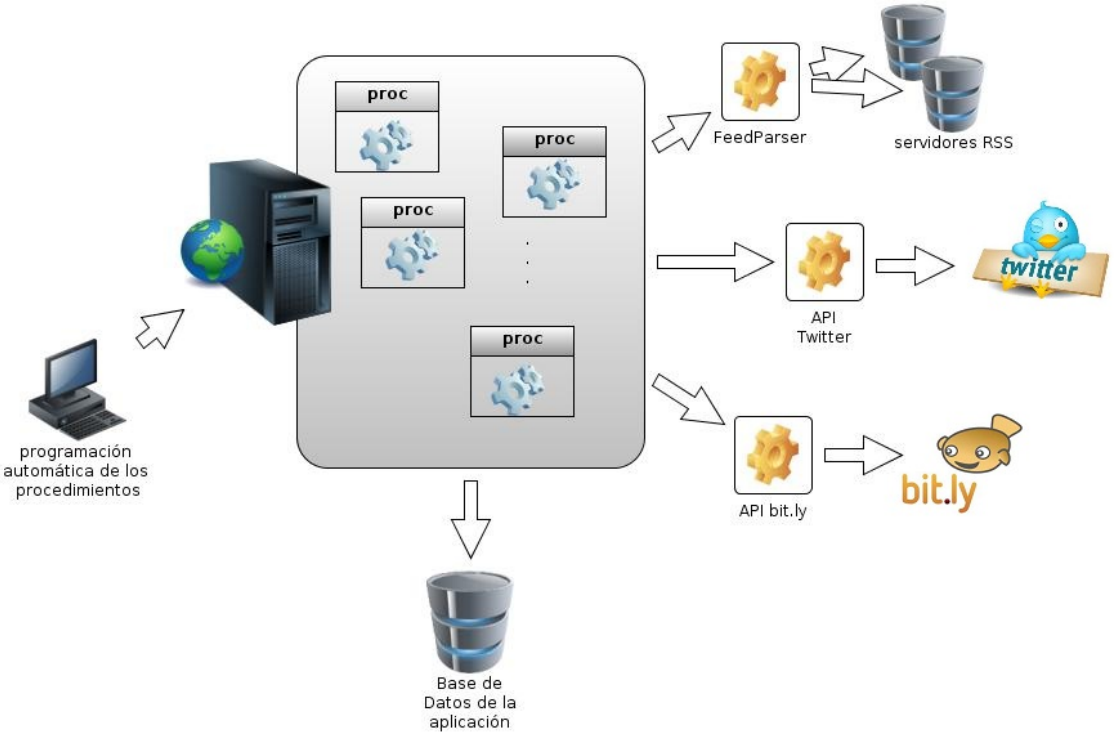


Figura 3.2: Solución gestión funcionalidades de los bots

Los procedimientos se encuentran albergados en un servidor Web. Cada procedimiento será encargado de ejecutar una de las tareas que el bot podrá realizar en Twitter. Para ello, los procedimientos serán capaces de comunicarse con el servidor de Base de datos para el almacenamiento/obtención/eliminación de datos necesarios para el correcto funcionamiento del bot, así como diversos APIs que proporcionan la comunicación de la aplicación con programas externos. Por este motivo serán necesarios:

- Un parseador de fuentes Rss: Este API permite a la aplicación obtener la información de las distintas fuentes Rss que serán leídas por el bot. Podrán ser fuentes Rss o Atom.
- El API de Twitter: Este API permite la comunicación entre la aplicación y la plataforma Twitter.

- El API de bit.ly: Este API nos permite la comunicación entre la aplicación y el servicio acortador de URLs de bit.ly. Dicha herramienta permite a la aplicación realizar acortamientos de las URLs publicadas por el bot en Twitter y obtener estadísticas de enlaces.

Existirá un mecanismo autómatico programable que ejecute cada uno de los procedimientos de manera periódica. Puesto que la plataforma de bots se encuentra instalada en una máquina Unix, utilizaremos el cron[1] del sistema para llevar a cabo esta tarea.

3.4. Tecnologías utilizadas

En esta sección se describen las distintas tecnologías empleadas para el desarrollo del proyecto.

3.4.1. Python

El lenguaje de programación con el que se han desarrollado los distintos componentes de la aplicación Web, así como el lenguaje en el que están programadas las distintas librerías externas empleadas por la aplicación, es Python.

Python es un lenguaje dinámico de programación, interpretado, extremadamente sencillo y que se utiliza en una amplia variedad de dominios de aplicación. Es un lenguaje de alto nivel. Suele ser comparado con otros lenguajes de programación con Tcl, Perl, Ruby o Java. Entre sus características distintivas se incluyen:

- Sintaxis sencilla y clara.
- Ofrece varios paradigmas: imperativo, orientado a objetos, funcional.
- Permite al desarrollador escribir sus códigos de manera potente y rápida.
- Gran cantidad de librerías estándar y módulos de terceros.
- Extensiones y módulos que permiten su fácil integración con otros lenguajes de programacion (Jython para Java, IronPython para .NET, etc.)
- Se encuentra disponible para la mayoría de sistemas operativos: Windows, Linux/Unix, OS/2, Mac, Amiga, entre otros.

[1]En el sistema operativo Unix, cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab.

- Es de código abierto. Cualquier persona es libre de descargar el intérprete de Python y estudiar su código, modificarlo y hacer su propio interprete.

Se ha elegido dicho lenguaje para la implementación del proyecto, puesto que permite mantener una sintaxis clara, compacta y ordenada, está disponible en la mayoría de los sistemas operativos y ante todo, permite hacer lo mismo que otros lenguajes de programación como Java, pero empleando la mitad de código y por tanto, la mitad de tiempo. Además posee de forma nativa todo el potencial de las expresiones regulares de otros lenguajes de programación como Perl.

3.4.2. Django

Django es un framework[2] Web escrito en Python. Dicho framework utiliza un patrón de diseño aproximado al **MVC**(**M**odelo-**V**ista-**C**ontrolador). El patrón MVC define una forma de desarrollar software en la que el código para definir y acceder a los datos (el modelo) se encuentra separado del pedido lógico de la asignación de ruta (el controlador), que a su vez se encuentra separado de la interfaz del usuario (la vista), permitiendo de esta forma que los componentes tengan un acoplamiento débil entre sí, es decir, cada pieza de la aplicación Web que funciona sobre Django tiene un único próposito clave y puede ser modificada independientemente sin afectar a las otras piezas.

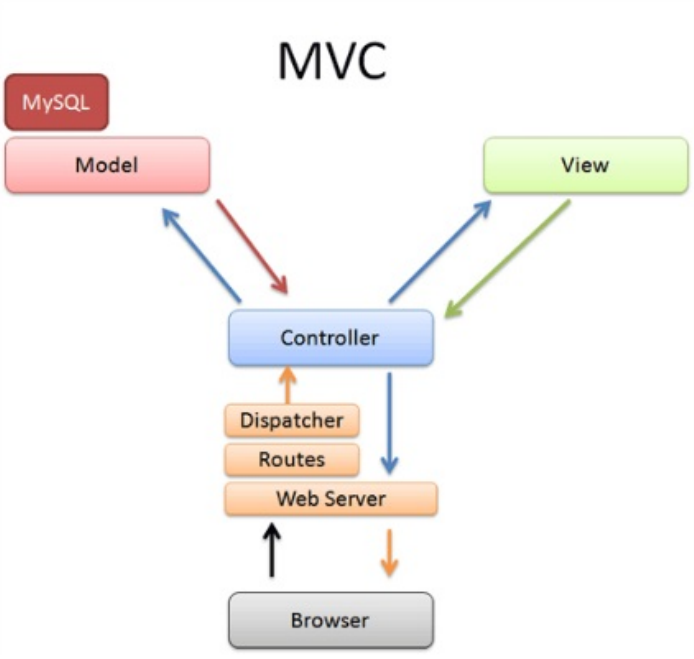


Figura 3.3: Patrón MVC

En Django el patrón MVC se separa de la siguiente manera:

[2]En desarrollo Web, un framework o entorno de desarrollo Web es un conjunto de herramientas que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web.
`http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf`

- Modelo: es la pieza dedicada al acceso a la base de datos, y es manejada por la capa de la base de datos de Django.
- Vista: es la pieza que selecciona qué datos mostrar y cómo mostrarlos. Es manejada por la vista y las plantillas.
- Controlador: es la pieza que delega a la vista dependiendo de la entrada del usuario. Es manejada por el propio framework llamando a la función adecuada de Python para la url obtenida

Para implementar dicho patrón, Django se basa en 4 archivos básicos para el desarrollo de cualquier aplicación:

- **models.py**: contiene una descripción de las tablas de la base de datos, como una clase Python. A esto se le llama el *modelo*. Usando esta clase de pueden crear, buscar, actualizar y borrar entradas de la base de datos usando código Python sencillo en lugar de escribir declaraciones SQL repetitivas.
- **views.py**: contiene la lógica de las distintas páginas de la aplicación Web. A cada función contenida en dicho fichero se la denomina *vista*.
- **urls.py**: especifica qué vista es llamada en función del patrón URL dado. Este archivo es muy importante porque indicará al framework a qué función(*vista*) deberá llamar cuando le llega una petición a una URL específica, y por ello se deberá tener especial cuidado al definirlo.
- **{template}.html**: son plantillas HTML que describen el diseño de las distintas páginas de las que consta la aplicación Web.

La aplicación ha sido desarrollada utilizando el framework Django, ya que dicho entorno de desarrollo Web permite construir aplicaciones Web potentes utilizando Python, de una manera correcta, sencilla, usando menos código que si no se utilizara dicho framework y por tanto mucho más rápido.

3.4.3. MySQL y MySQLdb

El sistema gestor de bases de datos utilizado es MySQL. Se tiene por tanto, una base de datos relacional. Una base de datos relacional representa un conjunto de datos que están almacenados en tablas entre las cuales se establecen unas relaciones para manejar los datos de una forma eficiente y segura. Para usar y gestionar una base de datos relacional se usa el lenguaje estándar de programación SQL. MySQL es Open Source, es decir, el código fuente se puede descargar y está accesible.

Se ha utilizado MySQL, de entre todos los gestores de Bases de Datos soportados por Django (PostgreSQL, SQLite, Oracle), ya que MySQL es altamente escalable y flexible, ofrece alto rendimiento, es robusto y posee alta disponibilidad. Además proporciona rapidez, seguridad y facilidad de manejo.

Para poder realizar operaciones sobre la base de datos MySQL y poder realizar conexiones con la misma desde Python, es necesaria la instalación del paquete **python-mysqldb**.

3.4.4. Apache y mod_python

El servidor HTTP es un servidor Web HTTP de código abierto para plataformas Unix, Windows y Mac, entre otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Apache presenta características altamente configurables, como Bases de Datos de autenticación y negociado de contenido. Además es fácil de instalar y de configurar.

Se ha utilizado el servidor Apache como gestor de peticiones Web puesto que es el servidor utilizado por Django (Apache 2.x). Igualmente para su instalación es necesario el módulo **mod_python**[3] (mod_python 3.x) que es el encargado de embeber el código Python en el servidor Apache y de cargarlo en memoria cuando se inicia el servidor, de tal forma que el código permanece en memoria a lo largo de la vida de un proceso de Apache.

Durante la instalación se utilizó el módulo **mod_python**, puesto que era el que se recomendaba por la página oficial de Django[4] en el momento que se realizó la instalación. Ahora lo desaconsejan a favor del módulo **mod_wsgi**[5] puesto que su implementación es más simple.

Para servir contenidos multimedia, también se ha utilizado el mismo servidor en el que se encuentra instalado Django. Desde el sitio Web de Django aconsejan utilizar un servidor Web separado para servir este tipo de contenidos si se van a servir grandes volúmenes de datos de contenidos multimedia, puesto que ralentizaría la parte de contenido estático de la aplicación. En este proyecto no se sirven grandes cantidades de contenido multimedia, por lo que se ha utilizado el mismo servidor para facilitar el proceso de instalación. Basta con desactivar el módulo mod_python para la gestión de este tipo de contenidos.

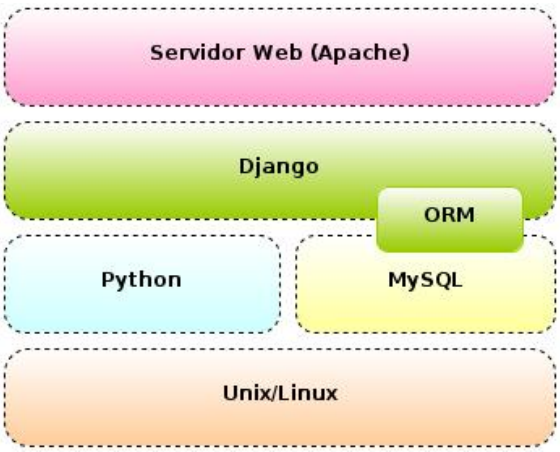


Figura 3.4: Diagrama de capas de Django

[3] http://www.modpython.org/
[4] https://www.djangoproject.com/
[5] http://code.google.com/p/modwsgi/

3.4.5. FeedParser

Universal Feed Parser [6] es un módulo de Python que permite descargar y parsear fuentes sindicadas. Permite manejar fuentes de tipo RSS, Atom y CDF. Para instalar dicho módulo es necesario Python 2.1 o superior.

Se utilizará dicho módulo para el parseo de fuentes sindicadas que podrán poseer los distintos bots creados por los usuarios de la plataforma de creación de bots, para que estos puedan publicar la información que se considere interesante de dichas fuentes.

3.4.6. API de Bit.ly

En la web existen varios servicios para acortar las URL, esto es, dada una URL cualquiera permite obtener otra URL de menor tamaño, que redirige hacia la URL original de forma transparente para el usuario. Uno de estos servicios web se llama **bit.ly**. Dicho acortador, entre otras funcionalidades, permite conocer el número de clicks que se han realizado sobre una determinada URL en un intervalo de tiempo. Es un acortador interesante puesto que proporciona un API para poder utilizar su servicio desde otros proyectos.

Como ya se ha visto, Twitter es un servicio de microblogging, que permite realizar publicaciones de 140 caracteres como máximo. Si dentro de una publicación de Twitter se inserta una URL ocurre muchas veces que el tamaño de esas URL es tan grande en caracteres que se consume la mayoría del espacio disponible para enviar texto. Por ello, lo que se suele hacer es acotar la URL, con lo que se puede insertar esa URL corta ganando espacio para escribir el resto del texto de la publicación.

Para poder acceder al API de bit.ly es necesario estar registrado en su plataforma Web[7]. Cuando se realiza el registro se proporciona un nombre de usuario (elegido por el usuario) y una llave de acceso (*token*) al API, que es una cadena de caracteres. Dicha llave de acceso permitirá a la plataforma de creacción de bots realizar peticiones al API de bit.ly, cuyo servicio devolverá sus respuestas en formato JSON[8] o XML.

La aplicación de creación de bots utilizará el API de bit.ly para acortar las direcciones URLs de las noticias de las fuentes sindicadas que utilizará el bot para realizar publicaciones en Twitter. Igualmente se utilizará el API para obtener estadísticas de los enlaces que puedan ser publicados por el bot o bien que hayan sido publicados por los distintos ***Followings*** del bot, para discriminar los enlaces interesantes de los que no lo son.

[6] `http://feedparser.org/`

[7] `http://bit.ly/`

[8] JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

3.4.7. API de Twitter

La plataforma de Twitter pone a disposición de sus usuarios un API que permite utilizar sus servicios desde otros proyectos. Twitter ofrece tres APIs: **Streaming API**, **REST API** y **Search API** aplicables a necesidades diferentes.

- El **Streaming API** proporciona un subconjunto de Tweets en casi tiempo real. Se establece una conexión permanente por usuario con los servidores de Twitter y mediante una petición HTTP se recibe un flujo continúo de Tweets en formato JSON. Se puede obtener una muestra aleatoria (statuses/sample), un filtrado (statuses/filter) por palabras claves o por usuarios. Sin embargo, los métodos más interesantes, como por ejemplo obtener todo el caudal de Tweets (statuses/firehose), obtener sólo los Tweets que tienen enlaces (statuses/links) o bien sólo los Tweets con ReTweets (statuses/retweet) no es un servicio que esté disponible para todo el mundo.

- El **Search API** suministra los Tweets con una antiguedad de hasta 7 días que se ajustan a la query solicitada. Es posible filtrar por: cliente utilizado, lenguaje y localización. No requiere autenticación y los tweets se obtienen en formato JSON o Atom. Este API ofrece una información más limitada del Tweet.

- El **REST API** ofrece a los desarrolladores el acceso al core de los datos de Twitter. Todas las operaciones que se pueden hacer a través de la plataforma Web es posible realizarlas desde dicho API. Dependiendo de la operación requiere o no autenticación, con el mismo criterio que en el acceso Web. Soporta los formatos: XML, JSON, RSS y Atom.

En el **Search API** y en el **REST API** existe una limitación de 150 peticiones a la hora por usuario o por IP si la petición no está autenticada, mientras que si la petición está autenticada mediante OAuth 3.4.8 el límite es de 350 peticiones/hora [Con10b].

Para poder realizar peticiones al API de Twitter de manera autenticada, es necesario dar de alta a la aplicación que vaya a hacer uso de dicho API en la página de aplicaciones de Twitter[9]. Una vez realizado el registro de la aplicación, se proporcionarán unas claves que permitirán el acceso al API de Twitter a dicha aplicación (**Consumer key** y **Consumer secret**).

Para realizar el proyecto ha sido necesario registrar una aplicación llamada **TweetyBots** en la página Web de aplicaciones de Twitter. Una vez obtenidas las claves del API de Twitter, la aplicación hará uso de dicho API para obtener la información necesaria para el correcto funcionamiento de los bots.

[9] `https://dev.twitter.com/apps`

Tweepy

Tweepy[10] es una librería de Python que permite hacer uso del API de Twitter. Permite acceder a las tres APIs de Twitter e incluye soporte para el protocolo de autenticación OAuth.

En este proyecto se utilizará la librería de Tweepy para todas las comunicaciones con el API de Twitter, al ser la más sencilla y la más completas de las librerías existentes para tal fin.

3.4.8. OAuth

OAuth es un protocolo abierto que permite autorización segura de un API de modo estándar y simple para aplicaciones de escritorio, móviles y Web. Con la identificación mediante OAuth se permite a una aplicación realizar publicaciones en una plataforma por mediación de un usuario sin tener que introducir la contraseña.

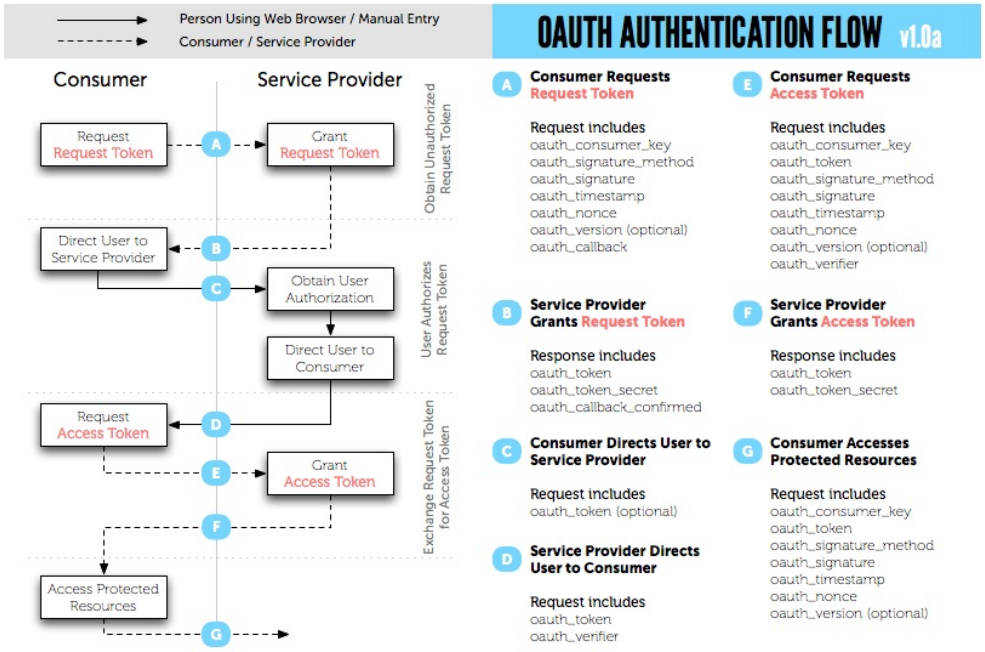


Figura 3.5: Diagrama de OAuth

El API de Twitter hace uso de este protocolo de autenticación para permitir que una aplicación registrada en su Web, pueda realizar acciones en Twitter por medio de un usuario que previamente haya dado su consentimiento a dicha aplicación para hacerlo. Para ello, cuando un usuario de Twitter desea que una aplicación pueda acceder a la gestión de su cuenta, se le redirige a la página OAuth de Twitter mostrada en la Figura 3.6 , en la que el usuario da su consentimiento. Una vez obtenido el consentimiento por parte del usuario, se le cederán a la aplicación las claves del usuario necesarias (**access_token_key** y **access_token_secret**) que le permitirán realizar acciones en Twitter por mediación del

[10] `http://pypi.python.org/pypi/tweepy/1.7.1`

usuario sin necesidad de pedir contraseña.

Figura 3.6: Página de confirmación consentimiento OAuth

En este proyecto, el soporte para la utilización del protocolo de autentificación OAuth para realizar comunicaciones seguras con el API de Twitter lo propociona la librería **Tweepy** 3.4.7

Capítulo 4

Análisis de la aplicación TweetyBots

4.1. Introducción

El objetivo general de este proyecto es la creacción de usuarios robotizados en Twitter de apariencia humana, ya que no se desea que sufran el rechazo de las personas a las que siguen, para analizar su evolución de popularidad de acuerdo a los distintos perfiles de comportamiento observados en Twitter.

Para que los bots simulen el comportamiento humano en la red social de Twitter, es necesario indicar una serie de parámetros de configuración. Dichos parámetros deberán ser configurados y modificados por el usuario, y ser almacenados de forma persistente para poder ser consultados y modificados por los distintos programas automatizados que simulen el comportamiento de los usuarios humanos de Twitter.

Puesto que las propiedades que definen el comportamiento de los distintos bots podrán cambiar a lo largo de la existencia del mismo, y deberán ser accesibles desde los programas automatizados, encargados de simular las distintas actividades que un usuario humano realiza habitualmente en la red de microblogging, se ha decidido almacenar estas propiedades de un modo persistente y accesible en una base de datos.

Igualmente, puesto que estas propiedades deberán ser configuradas y modificadas por los distintos usuarios que estén interesados en la creacción de bots sociales en Twitter, se ha decidido que estas propiedades se puedan visualizar y modificar por medio de una interfaz Web, cuyo objetivo es el de facilitar la tarea de modificación e inserción de datos a los distintos usuarios de la plataforma.

Para el desarrollo de los programas automátizados que simulen las distintas actividades que un usuario real puede realizar en Twitter, se ha decido crear un conjunto de funciones o procedimientos, escritos en código Python, que representarán cada una de las tareas que realizará el bot en Twitter.

4.2. Parámetros de configuración de los bots

A continuación se detallan las propiedades que definen el comportamiento de un bot, y que podrán cambiar a lo largo de su existencia.

4.2.1. Identidad

La identidad que define un bot, vendrá dada por la información de su perfil en Twitter: su nombre, su biografía, su localización, su página Web y su imagen del perfil.

4.2.2. Fuentes de Información

Las fuentes de información que permiten al bot realizar publicaciones en Tweet, son:

- Lista de Hashtags favoritos.
- Lista de medios de comunicación favoritos (fuente RSS).
- Lista de blogs favoritos (fuente RSS).
- Lista de citas.

4.2.3. Aficiones

Las aficiones del bot podrán clasificarse en: *Bricolaje*, *Cine*, *Cocina*, *Deportes*, *Idiomas*, *Juegos*, *Libros*, *Moda*, *Motor*, *Museos*, *Música*, *Teatro*, *Tecnología* y *Viajes*.

4.2.4. Lado social

El lado social vendrá dado por las distintas estrategias de comportamiento. A continuación se detallan cada una de las estrategias consideradas para que los distintos bots sean capaces de simular el comportamiento humano.

Estrategia de Following

La estrategia de Following define el modo en que el bot seguirá a otros usuarios de la red de Twitter. En dicha estrategia se define:

- El número máximo de Following que seguirá el bot al cabo del día.
- El perfil que han de cumplir los usuarios a los que el bot comience a seguir en Twitter (***perfil de Following***). Dicho perfil vendrá dado por:
  - El idioma en el que escriban los usuarios (Español, Inglés o ambos)
  - El número mínimo de Followers que tendrán los usuarios.
  - El porcentaje de Followers/Following máximo y mínimo que han de cumplir los usuarios.

    - El número mínimo de Menciones en los últimos 7 días que han de tener los usuarios.
    - El número mínimo de ReTweets en los últimos 7 días que han de tener los usuarios.

- Las condiciones que se han de cumplir para que el bot comience a seguir a un usuario en Twitter:

    - Por uso de Hashtags favoritos del bot (sí/no).
    - Por realizar ReTweet de los mensajes publicados por el bot (sí/no).
    - Por realizar Menciones al bot (sí/no).

  Igualmente, el bot siempre comenzará a seguir a aquellos Followers que cumplan el perfil de Following.

- Las condiciones que se han de cumplir para que el bot deje de seguir a un usuario en Twitter:

    - Por dejar de cumplir el perfil de Following (sí/no).
    - Si no corresponde al Follow realizado por el usuario en un cierto tiempo T (sí/no).

Estrategia de cortesía

La estrategia de cortesía define el modo en que el bot agradece los cumplidos que el resto de usuarios de Twitter realicen al bot. Dicha estrategia define si el bot:

- Enviará mensajes de agradecimiento a los usuarios que realicen ReTweets de sus publicaciones, a través de su Timeline público (sí/no).
- Enviará mensajes de agradecimiento a los usuarios que mencionen al bot en mensaje de tipo #FF, a través de su Timeline público (sí/no).
- Corresponderá a mensajes de tipo #FF que mencionen al bot, a través de su Timeline público.
- Enviará mensajes de agradecimiento a nuevos Followers a través de mensaje directo (DM) (sí/no).

Estrategia de personalidad

La estrategia de cortesía define el tipo de personalidad que desarrollará el bot a la hora de comunicarse a través de su Timeline. Dicha estrategia define si el bot:

- Enviará mensajes de "Buenos días" (sí/no), y si lo hará de manera personalizada o de forma general.
- Enviará mensajes de "Buenas noches" de forma generalizada (sí/no).
- Enviará mensajes que informen sobre qué está comiendo a sus seguidores (sí/no).

- Enviará mensajes que informen sobre actividades que está realizando con la familia (sí/no).

- Enviará mensajes de tipo TGIF (Thanks God It's Friday) los viernes (sí/no).

Estrategia de Follow on Friday

La estrategia de Follow on Friday configura la forma en la que el bot implementará la estrategia de Follow on Friday los viernes. Dicha estrategia define:

- El número máximo de mensajes de tipo #FF que publicará el bot los viernes.

- El número máximo de menciones que se realizarán en un mismo mensaje de tipo #FF.

- El perfil que han de cumplir los usuarios a los que el bot mencione en un mensaje de tipo #FF (**perfil de #FF**), que vendrá dado por:

  - El número mínimo de Followers que han de tener los usuarios.
  - El porcentaje de Followers/Following máximo y mínimo que han de cumplir los usuarios.
  - El número mínimo de Menciones en los últimos 7 días que han de tener los usuarios.
  - El número mínimo de ReTweets en los últimos 7 días que han de tener los usuarios.

Estrategia de Publicación

La estrategia de publicación define la frecuencia con la que el bot realizará sus publicaciones en Twitter. Dicha estrategia define:

- El número máximo de Tweets que el bot publicará a lo largo del día.

- El número máximo de ReTweets que el bot publicará a lo largo del día.

- El horario de publicación:

  - El horario de publicación en los días laborables (L-V).
  - El horario de publicación en los días festivos (S-D).
  - El horario de comida en el cual el bot podrá informar a sus Followers de lo que se encuentra comiendo.
  - La hora punta: En la que el bot enviará mensajes con mayor frecuencia.

4.3. Requisitos de funcionalidad de los distintos bots

La plataforma de bots, permitirá crear bots sociales que se comuniquen en la red de Twitter y establezcan relaciones con sus usuarios.

A continuación se exponen los requisitos de funcionalidad de los distintos bots. En dichos requisitos se define el conjunto de acciones que podrán realizarán en Twitter los distintos bots dados de alta por la plataforma **TweetyBots** así como los criterios que se han de cumplir para que el bot establezca relaciones con el resto de usuarios de Twitter.

Para una mejor comprensión de las actividades que el bot realizará en Twitter, así como el establecimiento de relaciones con los usuarios de Twitter, se recomienda revisar el Apartado 2.4 de este documento.

El objetivo sería estructurar las actividades cotidianas que un usuario humano realiza en Twitter. Por un lado se tendrán en cuenta los eventos que llevarán a un bot a realizar publicaciones en la plataforma y por otro lado se encontrarán las distintas acciones que podrán ser realizadas por el bot en función de dichos eventos: seguir a un usuario (***Follow***), dejar de seguir a un usuario (***Unfollow***), hacer un ReTweet de un estado (RT), publicar un Tweet y enviar un mensaje directo (DM).

En la Figura 4.1 se muestran por un lado (izquierda) los eventos que se han considerado pueden darse y por otro lado (derecha) las acciones en la que desembocan dichos eventos.

A continuación se detallan los requisitos que establecen las funcionalidades de los bots:

4.3.1. Comprobar menciones realizadas al bot

El bot deberá ser capaz de obtener las menciones que le realicen el resto de usuarios de Twitter, y en función de la estrategia de Following configurada por el usuario, que establecerá si el bot seguirá a aquellos usuarios que cumplan el perfil de Following y que le realicen menciones, decidirá si comenzará a seguir o no a los usuarios que le hayan mencionado.

4.3.2. Comprobar ReTweets realizados al bot

El bot deberá ser capaz de obtener los ReTweets (RTs) de sus estados realizados por el resto de usuarios de Twitter, y en función de la estrategia de Following configurada por el usuario, que establecerá si el bot seguirá a aquellos usuarios que cumplan el perfil de Following y que realicen RTs de sus estados, decidirá si comenzará a seguir o no a los usuarios que hayan realizado ReTweet de alguno de sus estados.

Igualmente, atendiendo a la estrategia de Cortesía configurada por el usuario, que establecerá si el bot enviará mensajes de agradecimiento a aquellos usuarios que realicen ReTweets de sus estados, enviará un mensaje de agradecimiento a través de su ***Timeline*** público a aquellos usuarios que hayan realizado un ReTweet (RT) a alguno de sus estados.

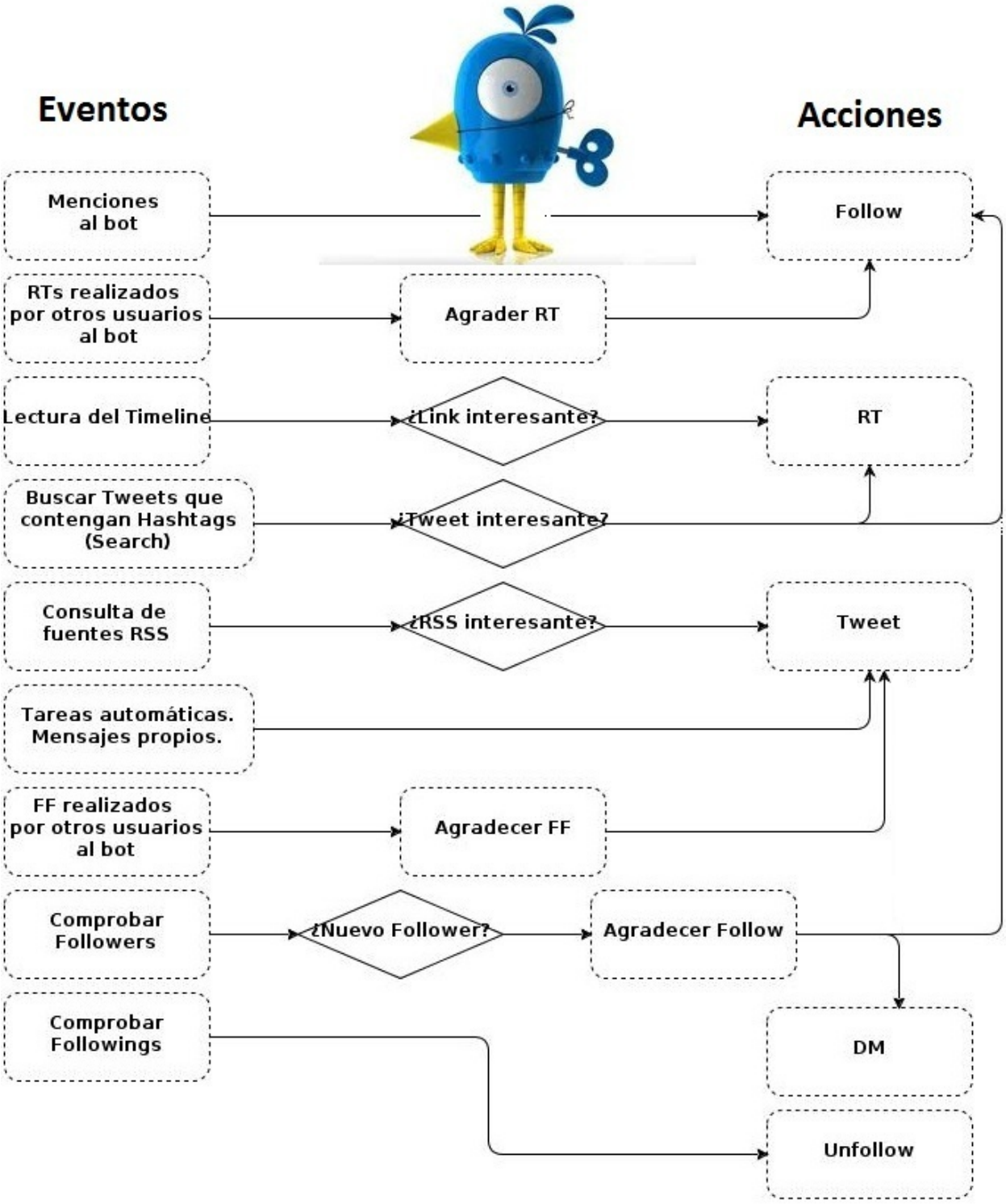


Figura 4.1: Esquema funcionalidades del bot

El bot deberá ser capaz de detectar los ReTweets realizados a sus estados por otros usuarios, tanto por el nuevo sistema de RT proporcionado por la plataforma Web de Twitter, como por el sistema tradicional (revisar Apartado 2.4.1).

4.3.3. Lectura del Timeline

El bot deberá ser capaz de leer su ***Timeline*** para obtener así los estados publicados por los usuarios de su lista de ***Following***. Una vez obtenidos dichos estados, el bot deberá ser capaz de discrimar aquellos estados que contengan URLs y que resulten de interés, para posteriormente y en función de la estrategia de Publicación configurada por el usuario, que establecerá el número máximo de ReTweets que podrá enviar el bot, decidirá si realizar o no un ReTweet de dichos estados.

Igualmente, el bot deberá ser capaz de recordar aquellos estados a los que haya realizado ReTweet (RT) para evitar publicar repetidas veces el mismo estado.

4.3.4. Buscar Tweets que contengan Hashtags

El bot deberá ser capaz de buscar estados de otros usuarios que contengan sus ***Hashtags*** favoritos a través del **Search API** de Twitter. Una vez obtenidos dichos estados, el bot deberá ser capaz de discrimar aquellos estados que resulten de interés, para posteriormente y en función de la estrategia de Publicación configurada por el usuario, que establecerá el número máximo de ReTweets que podrá enviar el bot, decidirá si realizar o no un ReTweet de dichos estados. El bot deberá ser capaz de recordar aquellos estados a los que haya realizado ReTweet (RT) para evitar publicar repetidas veces el mismo estado.

Igualmente, en función de la estrategia de Following configurada por el usuario, que establecerá si el bot seguirá a aquellos usuarios que cumplan el perfil de Following y que publiquen estados que contengan sus ***Hashtags*** favoritos, decidirá si comenzará a seguir o no a los usuarios que publiquen estados que contengan sus Hashtags favoritos.

4.3.5. Consultar fuentes RSS

El bot deberá ser capaz de parsear el contenido de sus fuentes RSS favoritas y discriminar aquellas fuentes que resulten de interés en función del número de clicks en un intervalo de tiempo con ayuda del **API de bit.ly** , para posteriormente, y en función de la estrategia de Publicación configurada por el usuario, que establecerá el número máximo de Tweets que podrá enviar el bot, decidirá si publicar o no el contenido de dicha fuente.

Igualmente, el bot deberá ser capaz de recordar aquellos estados que haya publicado para evitar publicar repetidas veces el mismo estado.

4.3.6. Tareas automáticas

El bot deberá ser capaz de publicar mensajes propios para imitar ciertos comportamientos que se dan entre los usuarios reales de Twitter. De este modo, en función de la estrategia de Personalidad definida por el usuario, el bot deberá de ser capaz de:

- Enviar mensajes de "Buenos Días" a sus seguidores a través de su ***Timeline*** público. Este tipo de mensajes se podrán enviar de manera personalizada (realizando menciones a seguidores) o de forma general.
- Enviar mensajes de "Buenas noches" a sus seguidores a través de su ***Timeline*** público de manera general.
- Informar a sus seguidores sobre qué está comiendo a través de su ***Timeline*** público, durante las horas establecidas en la estrategia de Publicación.
- Informar a sus seguidores sobre las tareas que se encuentra realizando con sus familiares a través de su ***Timeline*** público.
- Enviar mensajes de tipo ***TGIF (Thanks God It's Friday)*** los viernes a través de su ***Timeline*** público.
- Enviar mensajes de tipo ***FF (Follow on Friday)*** los viernes a través de su ***Timeline*** público, en función de la estrategia de Follow on Friday definida por el usuario, que establecerá el número máximo de mensajes de tipo #FF que el bot enviará los viernes a sus seguidores, así como el perfil que han de cumplir los usuarios a los que se enviarán este tipo de mensajes.

A la hora de realizar este tipo de publicaciones, el bot deberá tener en cuenta la estrategia de Publicación configurada por el usuario, que establecerá el número máximo de Tweets que podrá enviar el bot. Igualmente, el bot deberá ser capaz de recordar aquellos estados que haya publicado para evitar publicar repetidas veces el mismo estado.

4.3.7. Comprobar FF realizados al bot

El bot deberá ser capaz de detectar los #FF que le realicen el resto de usuarios de Twitter los viernes, y en función de la estrategia de Cortesía configurada por el usuario, que establecerá si el bot enviará o no mensajes de agradecimiento a aquellos usuarios que realicen #FF los viernes al bot, enviará mensajes de agradecimiento a través de su ***Timeline*** público a aquellos usuarios que le mencionen en un mensaje de tipo #FF. Igualmente, si la estrategia de Cortesía lo indica, el bot deberá corresponder a aquellos usuarios que le hayan mencionado en un mensaje de tipo #FF con un #FF de vuelta.

4.3.8. Comprobar lista de Followers

El bot deberá comprobar si se ha añadido un nuevo Follower a la lista de ***Followers*** del bot. Si se ha añadido un nuevo Follower a la lista y en función de la estrategia de Following configurada por el usuario, que establece el perfil que han de cumplir los nuevos Following

del bot, el bot deberá seguir al nuevo Follower si cumple el perfil de Following.

Igualmente, y en función de la estrategia de Cortesía configurada por el usuario, que indica si el bot enviará mensajes de agradecimiento a los nuevos Followers que cumplan el perfil de Following a través de mensaje directo (DM), el bot enviará un mensaje directo (DM) de agradecimiento a aquellos nuevos Followers que cumplan el perfil de Following.

4.3.9. Comprobar lista de Following

El bot deberá comprobar la lista de Following y en función de la estrategia de Following configurada por el usuario, que establece las circunstancias que deben darse para que el bot deje de seguir a un determinado Following, el bot deberá de dejar de seguir a un usuario si se cumplen las condiciones configuradas por el usuario que determinan que se ha de dejar de seguir a dicho usuario: porque el usuario ha dejado de cumplir el perfil de Following y/o porque no ha correspondido al bot en un tiempo determinado.

4.4. Requisitos de funcionalidad de la plataforma Web

Para el establecimiento de los parámetros de configuración, así como su modificación por los distintos usuarios, se creará una plataforma Web que sirva de interfaz entre la aplicación y los distintos usuarios.

A continuación se exponen los requisitos de funcionalidad de la plataforma Web.

4.4.1. Dar de alta a usuarios

La aplicación permitirá dar de alta a un nuevo usuario en la plataforma, almacenando de forma permanente sus datos (login y contraseña) en la Base de Datos.

4.4.2. Gestionar bots

La aplicación permitirá la gestión de los distintos bots registrados por un determinado usuario. Dicha gestión abarcará las funcionalidades de registrar un nuevo bot, para lo cual se almacenarán de forma persistente sus datos en la Base de Datos, visualizar los bots registrados en la aplicación por dicho usuario, modificar o dar de baja un bot.

4.4.3. Gestión de identidad del bot

La aplicación permitirá la gestión de la identidad de los distintos bots registrados por un determinado usuario. Dicha gestión abarcará las funcionalidades de visualizar y modificar la información del perfil de Twitter de un bot.

4.4.4. Gestión de las aficiones de un bot

La aplicación permitirá la gestión de las aficiones de los distintos bots registrados por un determinado usuario. Dicha gestión abarcará las funcionalidades de añadir, visualizar y borrar las aficiones asociadas a un bot.

4.4.5. Gestión de las fuentes RSS de un bot

La aplicación permitirá la gestión de las fuentes sindicadas de los distintos bots registrados por un determinado usuario. Dicha gestión abarcará las funcionalidades de añadir, visualizar y borrar las fuentes sindicadas asociadas a un bot.

Igualmente, la aplicación deberá proporcionar un mecanismo para añadir fuentes sindicadas a un bot a partir de una lista general de fuentes agrupadas en diversas categorías.

4.4.6. Gestión de los Hashtags de un bot

La aplicación permitirá la gestión de los Hashtags de los distintos bots registrados por un determinado usuario. Dicha gestión abarcará las funcionalidades de añadir, visualizar y borrar los Hashtags asociadas a un bot.

4.4.7. Gestión de la lista de Following de un bot

La aplicación permitirá la gestión de la lista de Following de los distintos bots registrados por un determinado usuario. Dicha gestión abarcará las funcionalidades de añadir un nuevo seguidor a la lista de seguidores de un bot, visualizar la lista de seguidores de un bot y borrar un usuario de la lista de seguidores de un bot.

4.4.8. Gestión de las estrategias de comportamiento de un bot

La aplicación permitirá la gestión de las estrategias de comportamiento de los distintos bots registrados por un determinado usuario. Dicha gestión abarcará las funcionalidades de visualizar y modificar las estrategias de comportamiento de un bot.

4.5. Requisitos de restricción de la plataforma Web

A continuación se exponen los requisitos de restricción de la plataforma Web.

4.5.1. Seguridad de la aplicación

Cualquier persona no puede acceder a la aplicación por lo que se ha de realizar un control de acceso al mismo. Para ello, por razones de seguridad, todos los usuarios que quieran

acceder a la aplicación tendrán que hacerlo mediante la introducción del login de usuario y la contraseña.

4.5.2. Fácil manejo

La aplicación ha de ser de fácil manejo para el usuario, ya que éste puede que no tenga experiencia en el entorno de la informática. Debe constar de una de una interfaz intuitiva para el usuario.

Además la aplicación se realizará de forma modular, para que ayude a la reutilización del código de la misma para posibles modificaciones futuras.

4.5.3. Lenguaje en castellano

El lenguaje de la aplicación será el castellano.

4.6. Casos de uso de la plataforma Web

En esta sección se exponen los casos de uso de la plataforma Web extraídos a partir de los requisitos de usuario. A continuación se detallan los casos de uso en forma de tablas.

Cuadro 4.1: Caso de Uso (I): Logearse

Nombre	***Logearse***
Actores	Usuario
Objetivo	Acceder al menú principal de la aplicación. Relacionado con el requisito 4.5.1.
Precondiciones	Ninguna
Escenario Básico	1. El usuario introduce su login y password. 2. El usuario presiona el botón "Entrar". 3. La aplicación carga la página principal y el menú correspondiente dependiendo de si el usuario tiene bots dados de alta o todavía no.

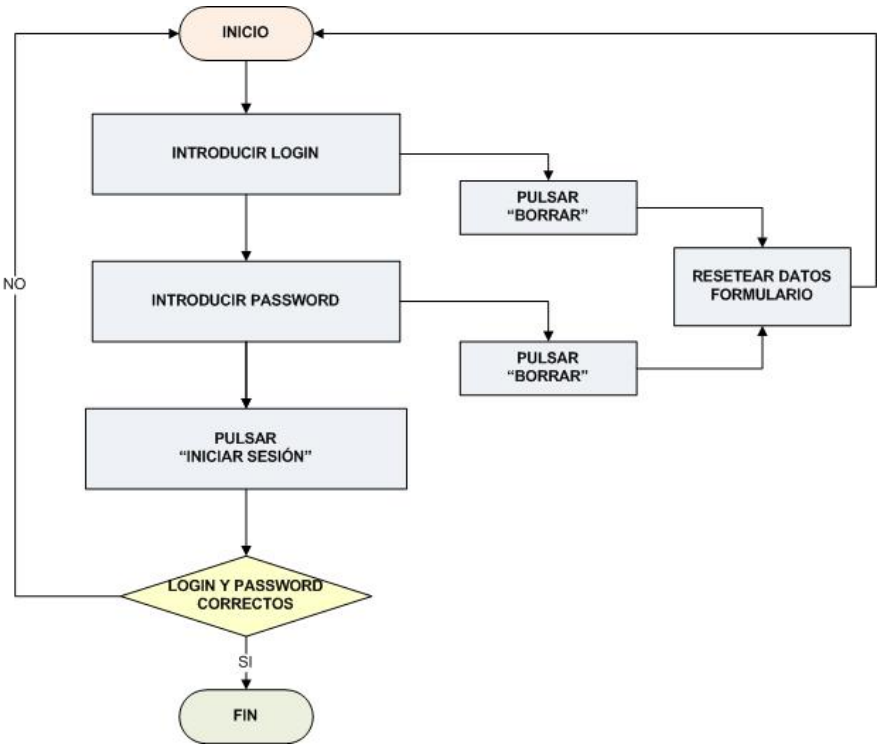


Figura 4.2: Diagrama de actividad: Logearse

Cuadro 4.2: Caso de Uso (II): Cerrar sesión

Nombre	***Logearse***
Actores	Usuario
Objetivo	Salir de la aplicación. Relacionado con el requisito 4.5.1.
Precondiciones	El usuario debe estar logueado en la aplicación previamente
Escenario Básico	1. El usuario hace click sobre el enlace "Cerrar sesión". 2. La sesión se cierra correctamente y la aplicación redirige al usuario a la pantalla de cierre correcto de sesión. Desde está página el usuario podrá nuevamente acceder a la página de login.

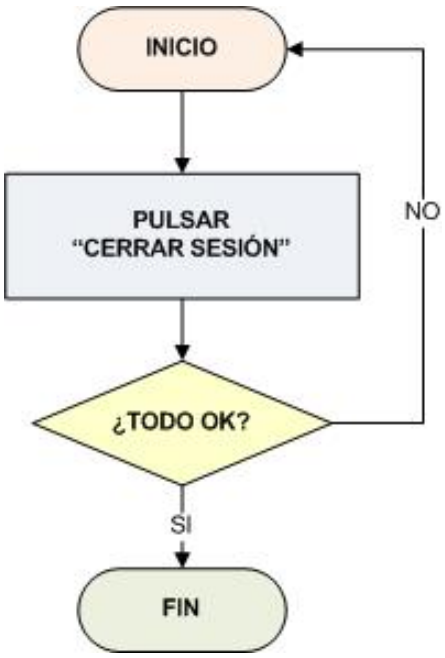


Figura 4.3: Diagrama de actividad: Cerrar sesión

Cuadro 4.3: Caso de Uso (III): Registrar un nuevo usuario

Nombre	***Registrar un nuevo usuario***
Actores	Usuario
Objetivo	Dar de alta a un usuario en la base de datos. Relacionado con el requisito 4.4.1.
Precondiciones	Ninguna
Escenario Básico	En la página de registro de la aplicación: 1. El usuario introduce login, contraseña y confirmación de contraseña. 2. El usuario pulsa el botón de "Enviar" para enviar los datos del formulario al servidor. 3. Si los datos son correctos (login único), se creará una nueva cuenta para el usuario y se le redirigirá a la página principal que le permitirá dar de alta un nuevo bot.

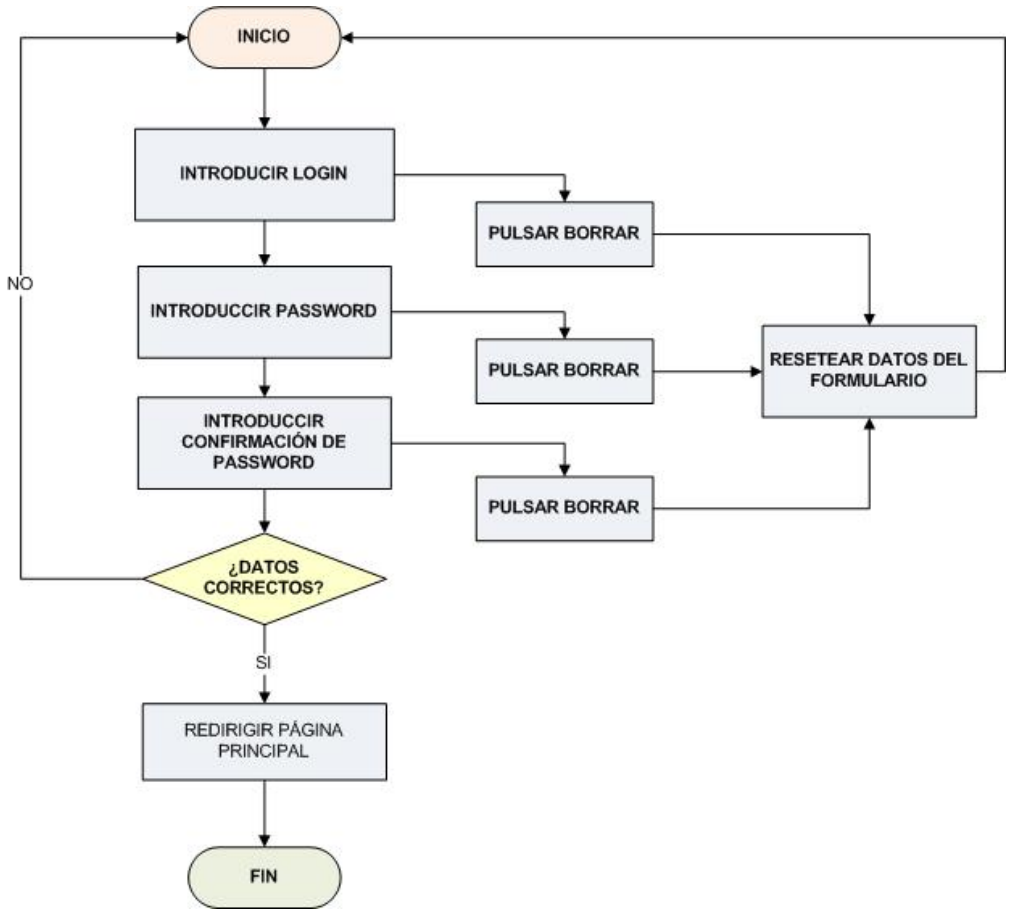


Figura 4.4: Diagrama de actividad: Registrar un nuevo usuario

Cuadro 4.4: Caso de Uso (IV): Registrar un nuevo bot

Nombre	***Registrar un nuevo bot***
Actores	Usuario
Objetivo	Crear un nuevo bot. Relacionado con el requisito 4.4.2.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la página principal de la aplicación: 1. En el menú de la izquierda, el usuario selecciona la opción "Registrar un nuevo bot". 2. El usuario pulsa el botón "Sign in with Twitter". 3. Se redirige el usuario a la página de autenticación de OAuth de Twitter para autorizar a la aplicación a que utilice su cuenta. 4. El usuario introduce login y password de Twitter y pulsa el botón "Autorizar la aplicación". 5. Se redirige al usuario a la página principal de la aplicación en la que se visualizan los bots que el usuario tiene dados de alta en la aplicación.

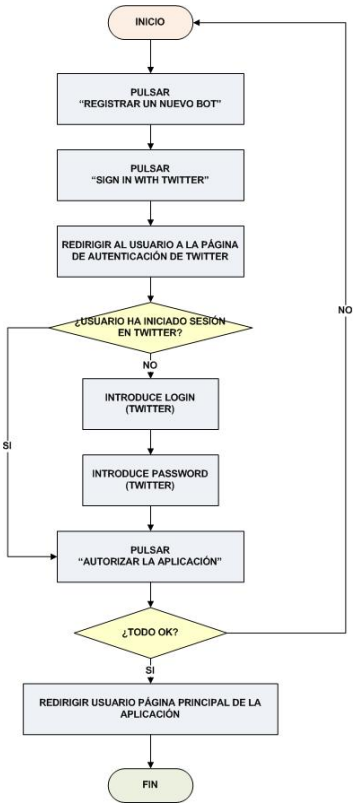


Figura 4.5: Diagrama de actividad: Registrar un nuevo bot

Cuadro 4.5: Caso de Uso (V): Visualizar bots

Nombre	***Visualizar bots***
Actores	Usuario
Objetivo	Visualizar los bots registrados por un usuario. Relacionado con el requisito 4.4.2.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la página principal de la aplicación: 1. En el menú de la izquierda, el usuario selecciona la opción "Visualiza tus bots". 2. Se redirige al usuario a la página principal de la aplicación en la que se visualizan los bots que el usuario tiene dados de alta en la aplicación.

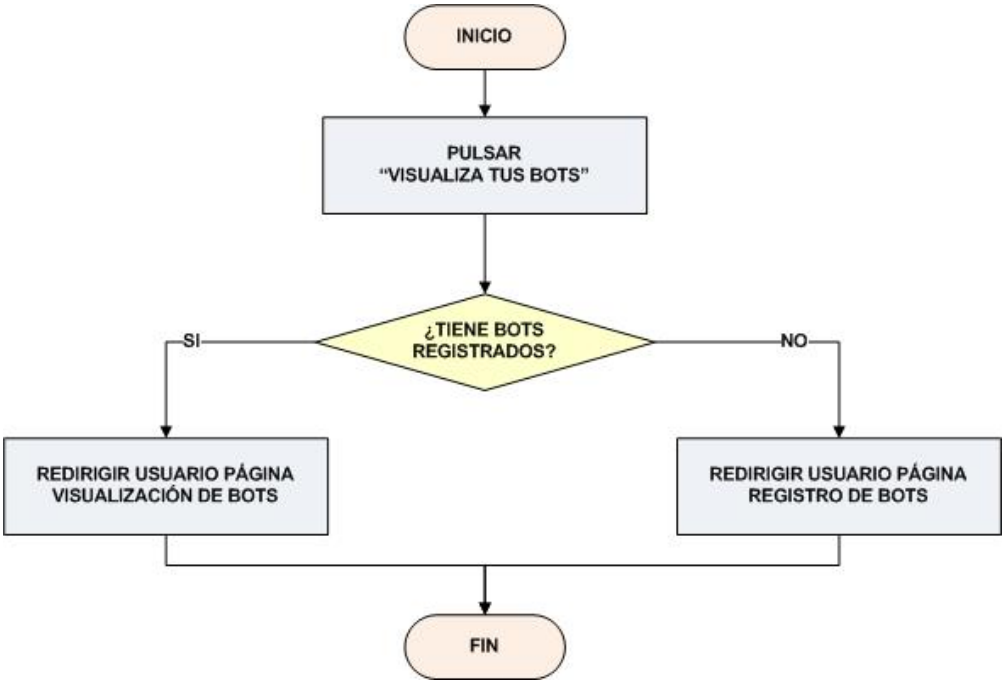


Figura 4.6: Diagrama de actividad: Visualizar bots

Cuadro 4.6: Caso de Uso (VI): Modificar bots

Nombre	***Modificar bots***
Actores	Usuario
Objetivo	Modificar las propiedades de un bot registrado por un usuario. Relacionado con el requisito 4.4.2.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la página principal de la aplicación en la que se visualizan los bots que el usuario tiene dados de alta en la aplicación: 1. El usuario pulsa en el botón "modificar" de un determinado bot. 2. Se redirige al usuario a la página principal de la aplicación en la que se permite modificar las propiedades de un bot.

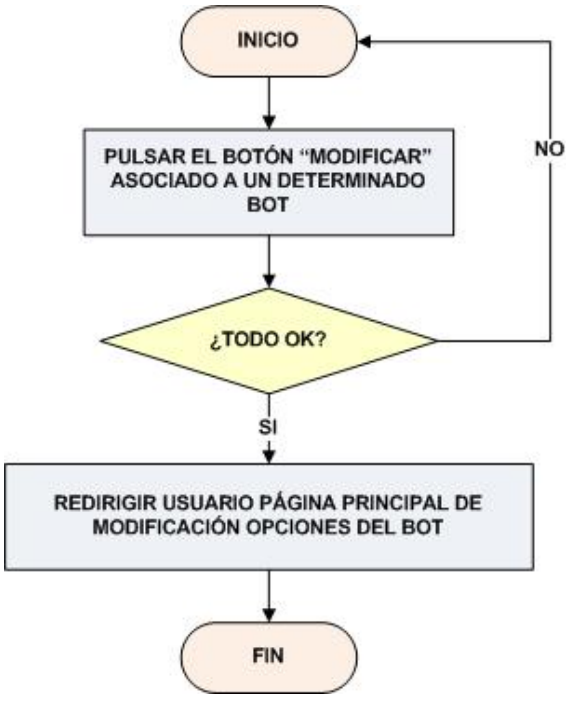


Figura 4.7: Diagrama de actividad: Modificar bots

Cuadro 4.7: Caso de Uso (VII): Dar de baja un bot

Nombre	***Dar de baja un bot***
Actores	Usuario
Objetivo	Dar de baja a un bot de la aplicación. Relacionado con el requisito 4.4.2.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la página principal de la aplicación: 1. En el menú de la izquierda, el usuario selecciona la opción "Dar de baja un bot". 2. El usuario selecciona un bot y pulsa el botón "borrar". 3. Se muestra un mensaje de confirmación antes de borrar definitivamente el bot de la aplicación. 4. Si el usuario confirma que desea eliminar al bot de la aplicación, se borrará toda la información disponible en base de datos asociada al bot. 5. Se redirige al usuario a la página principal de la aplicación en la que se permite dar de baja un bot de la lista de bots que el usuario tiene dados de alta en la aplicación.

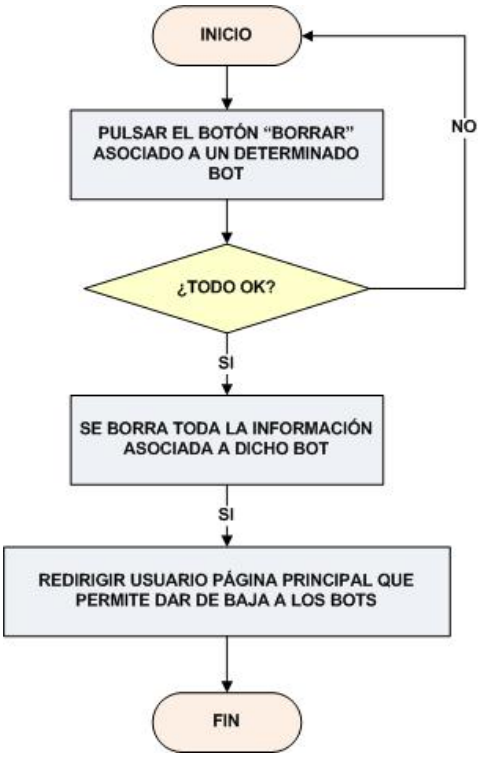


Figura 4.8: Diagrama de actividad: Dar de baja un bot

Cuadro 4.8: Caso de Uso (VIII): Gestionar perfil bot

Nombre	***Gestionar perfil bot***
Actores	Usuario
Objetivo	Modificar las propiedades del perfil de un bot registrado por un usuario. Relacionado con el requisito 4.4.3.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la página principal de la aplicación en la que se permiten modificar las propiedades de un bot que el usuario tiene dado de alta en la aplicación: 1. El usuario selecciona la pestaña "Información perfil". 2. Se muestra la información del perfil en Twitter del bot. 3. El usuario modifica algún campo y pulsa el botón "guardar". 4. El servidor recibe los datos, y envía la petición de modificación de datos de perfil a Twitter mediante REST API. 5. Se almacenan los cambios de manera persistente en el servidor de Base de Datos.

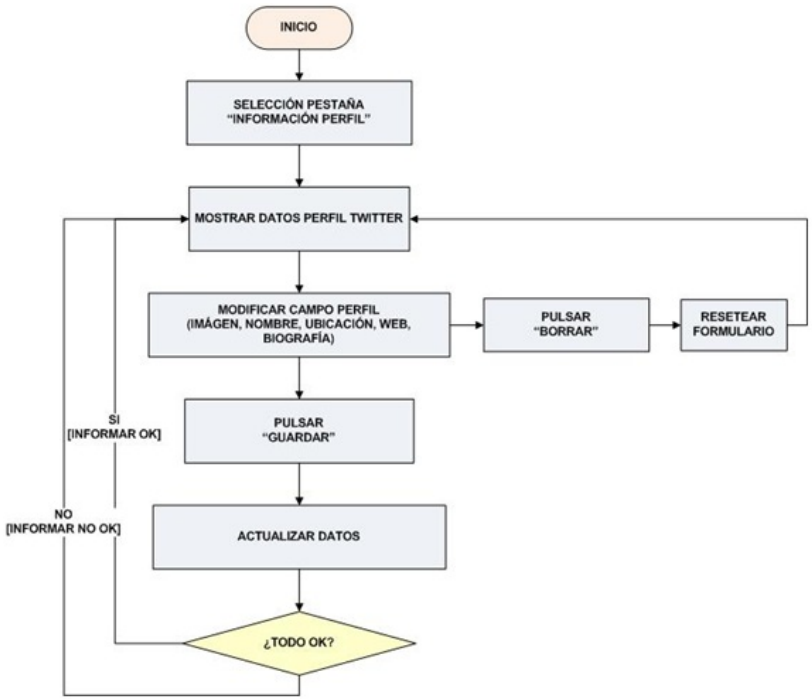


Figura 4.9: Diagrama de actividad: Gestionar perfil bot

Cuadro 4.9: Caso de Uso (IX): Añadir aficiones a un bot

Nombre	***Añadir aficiones a un bot***
Actores	Usuario
Objetivo	Modificar las aficiones asociadas a un bot registrado por un usuario. Relacionado con el requisito 4.4.4.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la página principal de la aplicación en la que se permiten modificar las propiedades de un bot que el usuario tiene dado de alta en la aplicación: 1. El usuario selecciona la pestaña "Aficiones". 2. Se muestran las aficiones asociadas a un bot. 3. El usuario selecciona una afición de la lista de aficiones disponibles y pulsa el botón "añadir". 4. El servidor recibe los datos. 5. Tras comprobar que la afición seleccionada no se encuentra ya asociada a dicho bot, almacena los cambios de manera persistente en el servidor de Base de Datos.

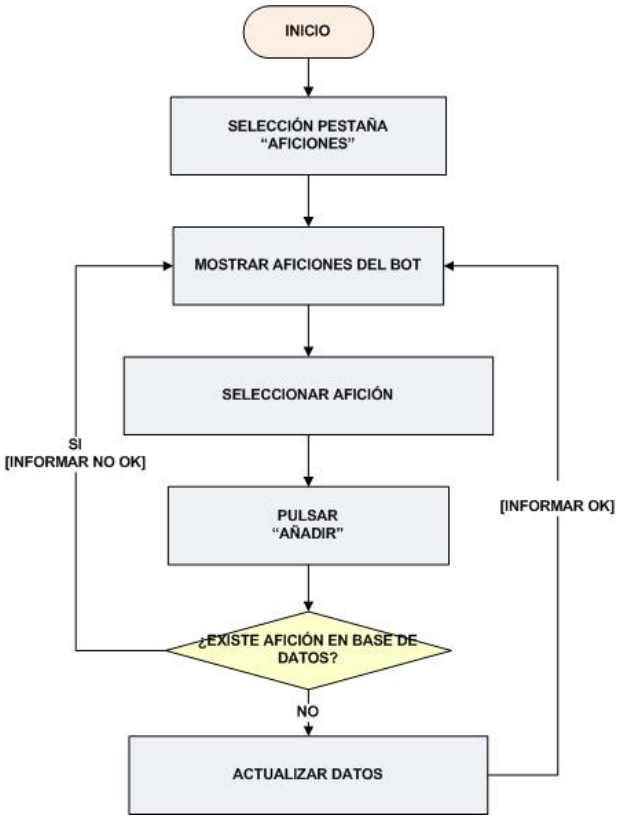


Figura 4.10: Diagrama de actividad: Añadir afición

Cuadro 4.10: Caso de Uso (X): Borrar aficiones a un bot

Nombre	***Borrar aficiones a un bot***
Actores	Usuario
Objetivo	Borrar una afición asociada a un bot registrado por un usuario. Relacionado con el requisito 4.4.4.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la página principal de la aplicación en la que se permiten modificar las propiedades de un bot que el usuario tiene dado de alta en la aplicación: 1. El usuario selecciona la pestaña "Aficiones". 2. Se muestran las aficiones asociadas a un bot. 3. El usuario selecciona una afición de la lista de aficiones asociadas al bot y pulsa el botón "eliminar". 4. El servidor recibe los datos. 5. Tras comprobar que la afición seleccionada se encuentra asociada a dicho bot, elimina el registro asociado de la Base de Datos.

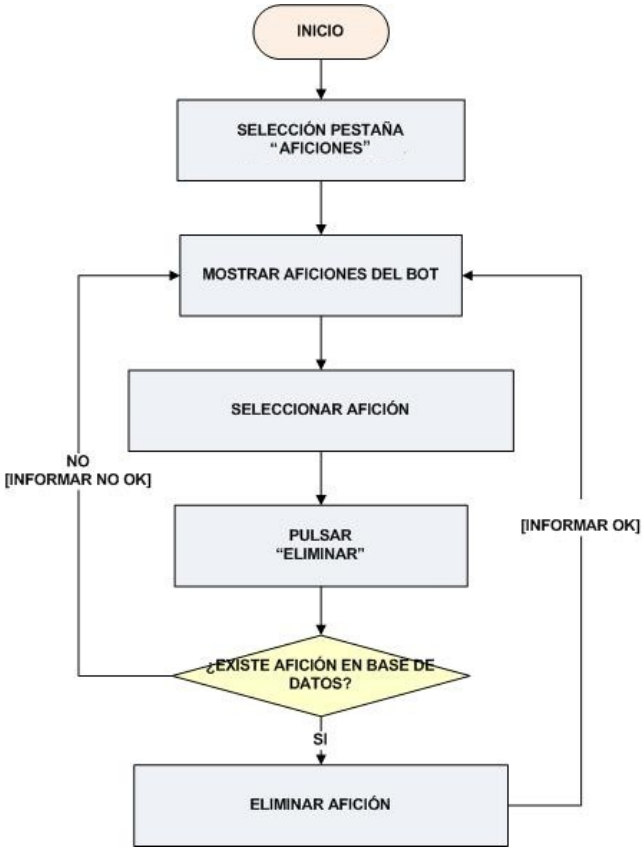


Figura 4.11: Diagrama de actividad: Borrar aficiones a un bot

Cuadro 4.11: Caso de Uso (XI): Añadir fuente RSS al bot

Nombre	***Añadir fuente RSS al bot***
Actores	Usuario
Objetivo	Añadir una fuente RSS a la lista de fuentes de un bot registrado por un usuario. Relacionado con el requisito 4.4.5.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la pestaña "Fuentes Rss favoritas": 1. Se muestra la lista de fuentes RSS asociadas a un bot. 2. El usuario introduce una nueva fuente. 3. El usuario introduce una categoría asociada a la fuente. 4. El usuario pulsa el botón "añadir". 5. El servidor recibe los datos.

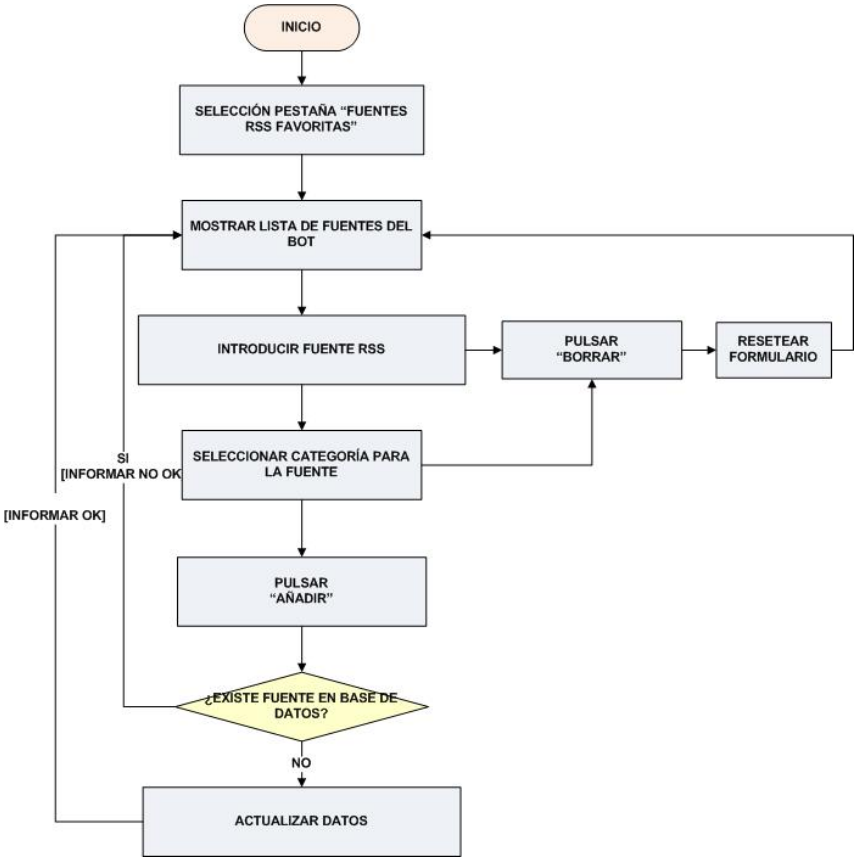


Figura 4.12: Diagrama de actividad: Añadir fuente RSS al bot

Cuadro 4.12: Caso de Uso (XII): Eliminar fuente RSS de un bot

Nombre	***Eliminar fuente RSS de un bot***
Actores	Usuario
Objetivo	Eliminar una fuente RSS de la lista de fuentes asociadas a un bot registrado por un usuario. Relacionado con el requisito 4.4.5.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la pestaña "Fuentes Rss favoritas": 1. Se muestra la lista de fuentes RSS asociadas a un bot. 2. El selecciona una fuente RSS de la lista de fuentes. 3. El usuario pulsa el botón "borrar". 4. El servidor recibe los datos. 5. Tras comprobar que la fuente Rss introducida se encuentra asociada a dicho bot, eliminará el registro correspondiente de la Base de Datos.

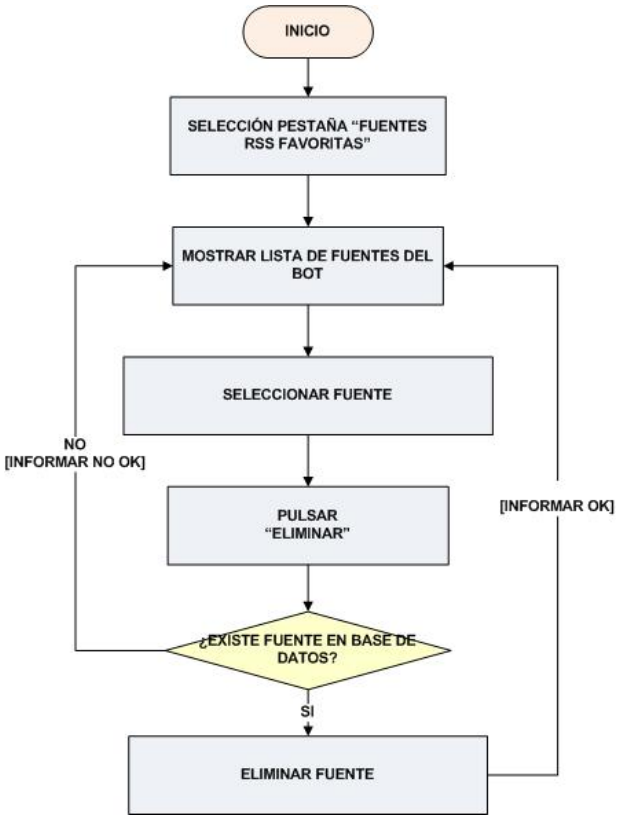


Figura 4.13: Diagrama de actividad: Eliminar fuente RSS de un bot

Cuadro 4.13: Caso de Uso (XIII): Añadir fuente RSS asociada a bot a la lista general

Nombre	***Añadir fuente RSS asociada a bot a la lista general***
Actores	Usuario
Objetivo	Añadir una fuente RSS de la lista de fuentes de un bot registrado por un usuario a la lista de fuentes generales de la aplicación. Relacionado con el requisito 4.4.5.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la pestaña "Fuentes Rss favoritas": 1. Se muestra la lista de fuentes RSS asociadas a un bot. 2. El usuario selecciona una fuente RSS de la lista de fuentes. 3. El usuario pulsa el botón "añadir fuente a lista". 4. El servidor recibe los datos. 5. Tras comprobar que la fuente Rss introducida no se encuentra en la lista de fuentes generales, almacena la información de manera persistente en la Base de Datos.

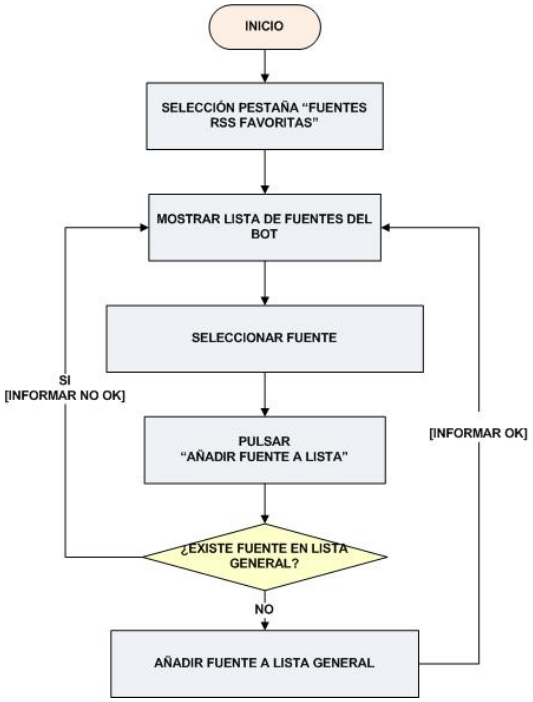


Figura 4.14: Diagrama de actividad: Añadir fuente RSS asociada a bot a la lista general

Cuadro 4.14: Caso de Uso (XIV): Añadir fuente RSS de la lista general a un bot

Nombre	***Añadir fuente RSS de la lista general a un bot***
Actores	Usuario
Objetivo	Añadir una fuente RSS de la lista de fuentes generales a la lista de fuentes Rss de un bot registrado por un usuario. Relacionado con el requisito 4.4.5.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la pestaña "Fuentes Rss favoritas": 1. El usuario pulsa "Ver fuentes recomendadas". 2. Se redirige al usuario a la página que muestra la lista de fuentes generales de la aplicación. 3. Se muestran la lista de fuentes RSS generales de la aplicación. 4. El usuario selecciona una fuente Rss de la lista de fuentes generales. 5. El usuario pulsa el botón "añadir fuente a favoritos". 6. El servidor recibe los datos 7. Tras comprobar que la fuente Rss introducida no se encuentra en la lista de fuentes asociadas al bot, almacena la información de manera persistente en la Base de Datos.

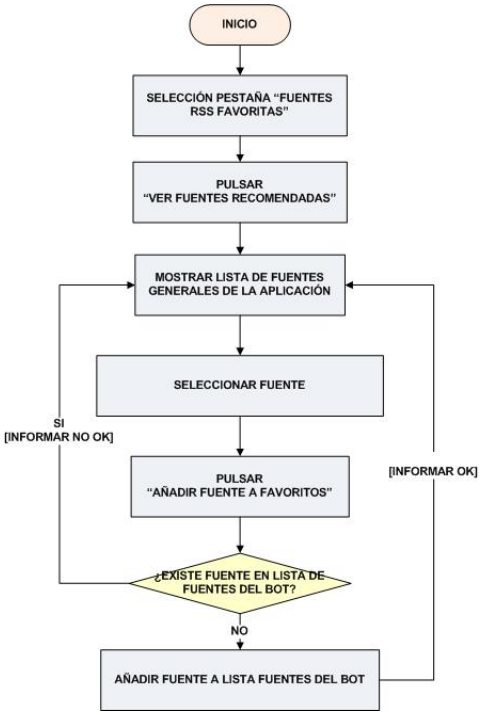


Figura 4.15: Diagrama de actividad: Añadir fuente RSS de la lista general a un bot

Cuadro 4.15: Caso de Uso (XV): Añadir Following a un bot

Nombre	***Añadir Following a un bot***
Actores	Usuario
Objetivo	Añadir un nuevo Following a la lista de Following de un bot registrado por un usuario. Relacionado con el requisito 4.4.7.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la pestaña "Escuchando a": 1. Se muestra la lista de Following del bot. 2. El usuario introduce el nombre de un usuario de Twitter y pulsa el botón "buscar". 3. Se muestra la información del perfil en Twitter del usuario introducido. 4. El usuario pulsa el botón "seguir". 5. El servidor recibe los datos. Tras comprobar que el usuario introducido no se encuentra en la lista de Following del bot, envía la orden de empezar a seguir a dicho usuario en Twitter mediante REST API. 6. Si todo va bien, almacena la información del nuevo Following de manera persistente en la Base de Datos.

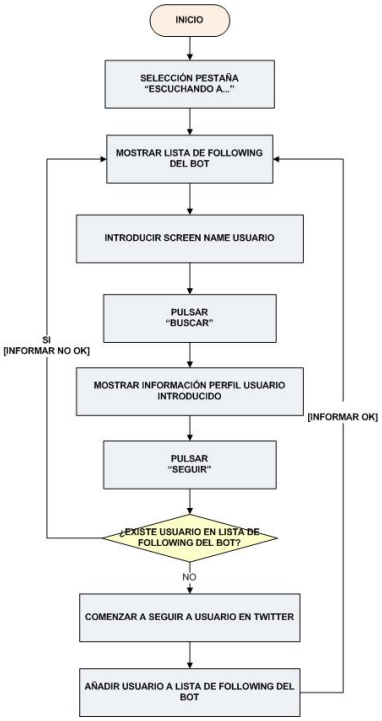


Figura 4.16: Diagrama de actividad: Añadir Following a un bot

Cuadro 4.16: Caso de Uso (XVI): Eliminar Following de un bot

Nombre	***Eliminar Following de un bot***
Actores	Usuario
Objetivo	Añadir un nuevo Following a la lista de Following de un bot registrado por un usuario. Relacionado con el requisito 4.4.7.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la pestaña "Escuchando a": 1. Se muestra la lista de Following del bot. 2. Seleccionar un Following de la lista de Following del bot. 3. El usuario pulsa el botón "eliminar usuario". 4. Tras comprobar que el usuario introducido se encuentra en la lista de Following del bot envía la petición de dejar de seguir en Twitter a dicho usuario mediante REST API. 5. Si todo va bien, se elimina el registro asociado a dicho usuario de la Base de Datos y se almacena al usuario en Lista negra.

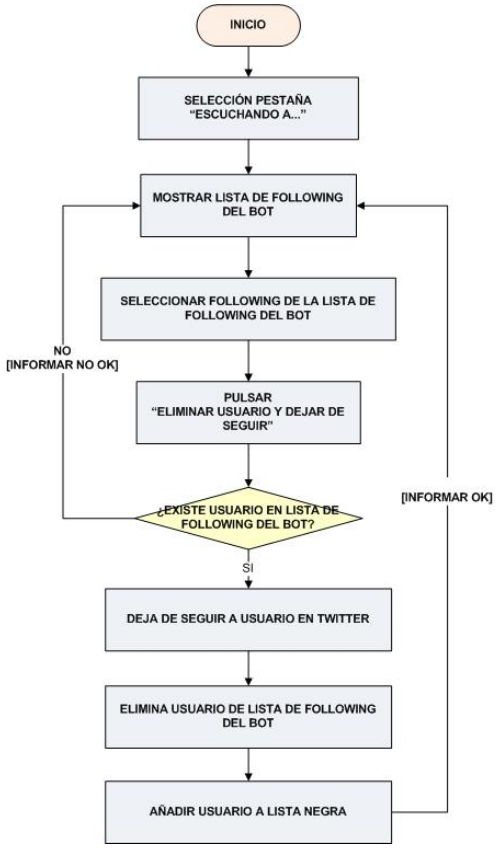


Figura 4.17: Diagrama de actividad: Eliminar Following de un bot

Cuadro 4.17: Caso de Uso (XVII): Añadir Hastag a un bot

Nombre	***Añadir Hastag a un bot***
Actores	Usuario
Objetivo	Añadir un nuevo Hashtag a la lista de Hashtags favoritos de un bot registrado por un usuario. Relacionado con el requisito 4.4.6.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la pestaña "Hastags favoritos": 1. Se muestra la lista de Hashtags favoritos del bot. 2. El usuario introduce un Hashtag. 3. El usuario pulsa el botón "añadir". 4. El servidor recibe los datos. 5. Tras comprobar que el Hashtag introducido no se encuentra en la lista de Hashtags favoritos del bot, almacena la información del nuevo Hashtag de manera persistente en la Base de Datos.

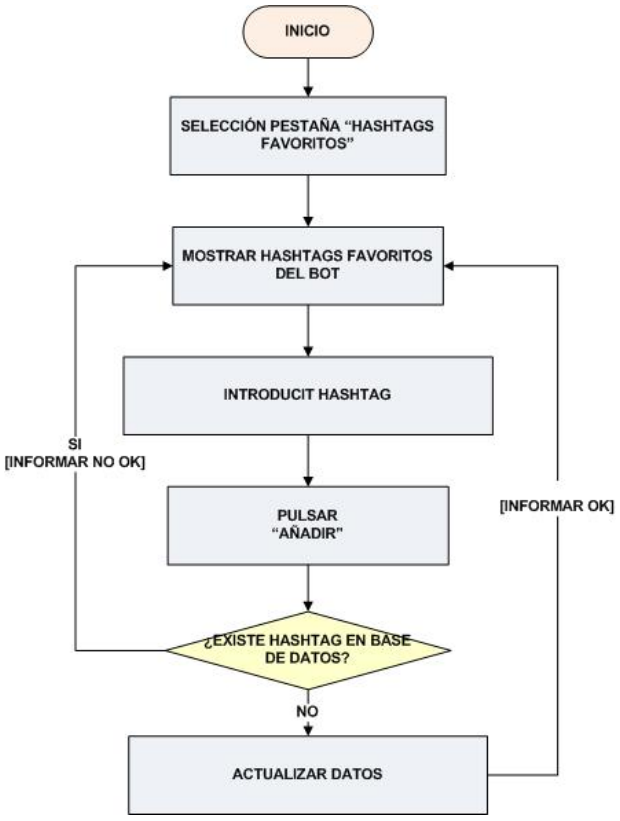


Figura 4.18: Diagrama de actividad: Añadir Hastag a un bot

Cuadro 4.18: Caso de Uso (XVIII): Eliminar hashtag de un bot

Nombre	***Eliminar hashtag de un bot***
Actores	Usuario
Objetivo	Eliminar un hashtag asociado a un bot registrado por un usuario. Relacionado con el requisito 4.4.6.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la pestaña "Hastags favoritos": 1. Se muestra la lista de Hashtags favoritos del bot. 2. El usuario selecciona un Hashtag de la lista de Hashtags asociados al bot y pulsa el botón "eliminar". 3. El servidor recibe los datos. 4. Tras comprobar que el hashtag seleccionado se encuentra asociado a dicho bot, elimina el registro asociado de la Base de Datos.

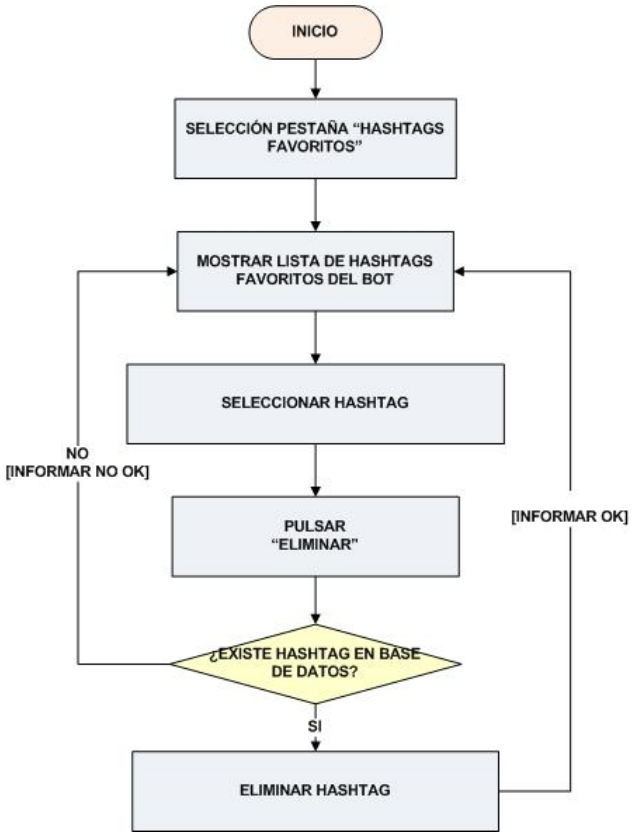


Figura 4.19: Diagrama de actividad: Eliminar hashtag de un bot

Cuadro 4.19: Caso de Uso (XIX): Gestión estrategias de un bot

Nombre	***Gestión estrategias de un bot***
Actores	Usuario
Objetivo	Visualizar y modificar las estrategias de comportamiento asociadas a un bot registrado por un usuario. Relacionado con el requisito 4.4.8.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la pestaña "Estrategias de comportamiento": 1. Se muestra la lista de estrategias de comportamiento asociados a un bot, junto con su configuración. 2. El usuario selecciona una estrategia de comportamiento de la lista. 3. El usuario pulsa el botón "modificar". 4. Se redirige al usuario a la página que permite modificar los parámetros de configuración de la estrategia seleccionada. 5. El usuario modifica algún parámetro de configuración. 6. El usuario pulsa el botón "guardar". 7. El servidor recibe los datos. 8. Tras comprobar que los datos introducidos son correctos, se actualiza el registro asociado de la Base de Datos.

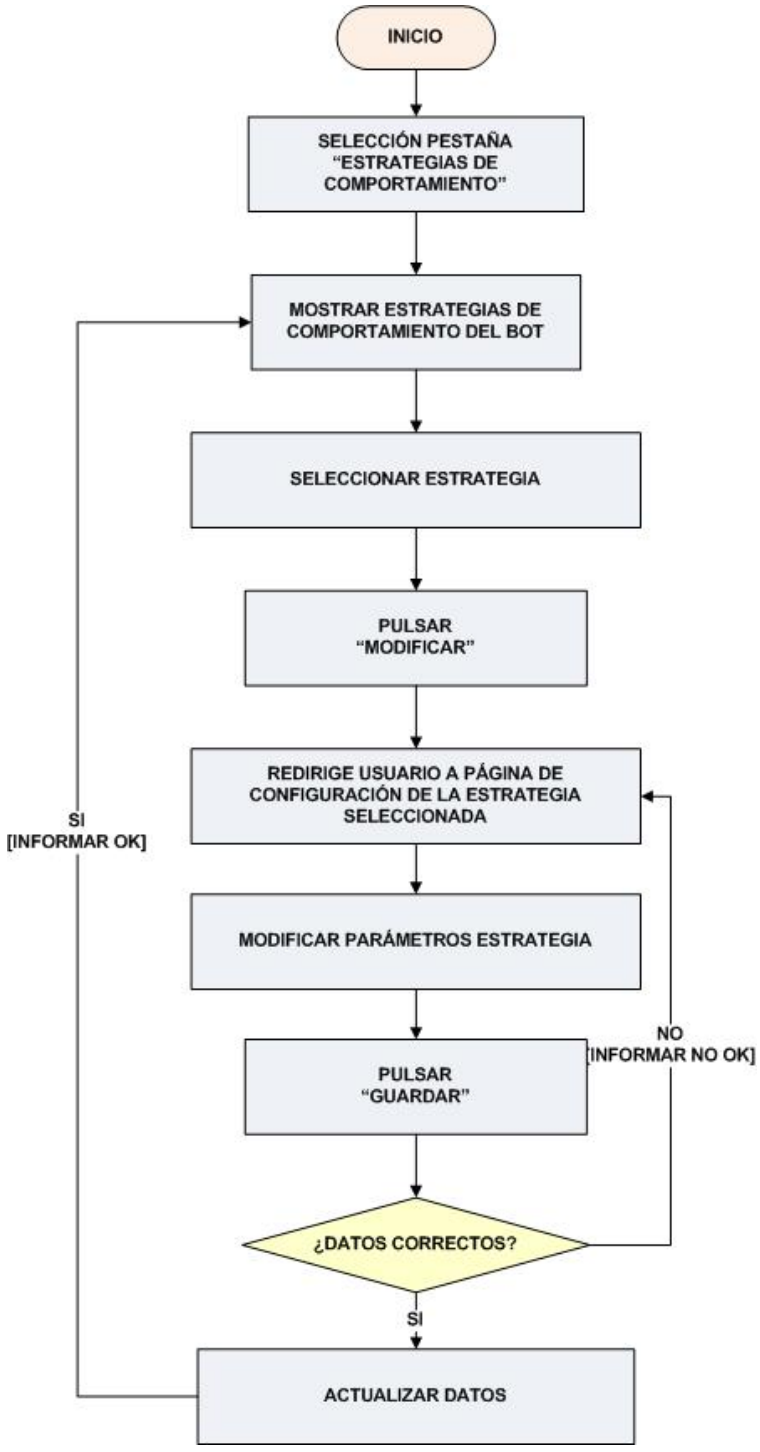


Figura 4.20: Diagrama de actividad: Gestión estrategias de un bot

Cuadro 4.20: Caso de Uso (XX): Eliminar configuración estrategias de un bot

Nombre	***Eliminar configuración estrategias de un bot***
Actores	Usuario
Objetivo	Eliminar la configuración asociada a las estrategias de comportamiento asociadas a un bot registrado por un usuario. Relacionado con el requisito 4.4.8.
Precondiciones	El usuario debe estar logueado en la aplicación previamente.
Escenario Básico	En la pestaña "Estrategias de comportamiento": 1. Se muestra la lista de estrategias de comportamiento asociados a un bot, junto con su configuración. 2. El usuario selecciona una estrategia de comportamiento de la lista. 3. El usuario pulsa el botón "eliminar". 4. El servidor recibe los datos. 5. Se resetea el valor de los parámetros de configuración de la estrategia seleccionada. 6. Se actualiza el registro asociado de la Base de Datos.

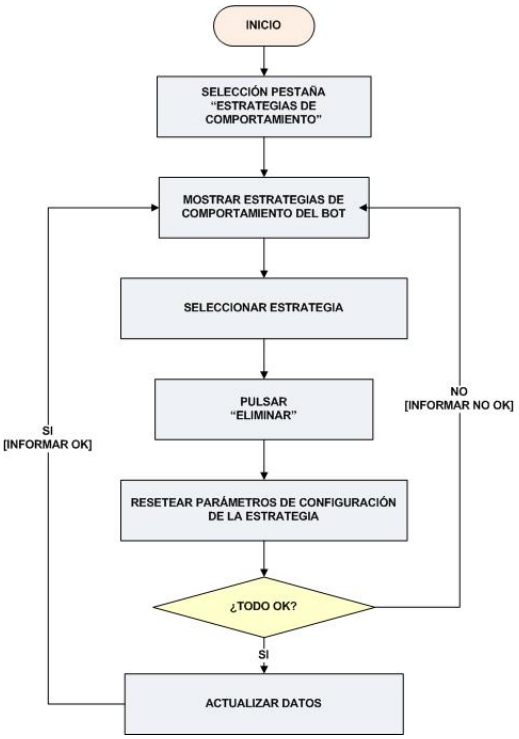


Figura 4.21: Diagrama de actividad: Eliminar configuración estrategias de un bot

4.6.1. Diagramas de casos de uso de la plataforma Web

Los diagramas de uso se han dividido en dos escenarios:

- Escenario 1: Corresponde a las funcionalidades que puede realizar en la plataforma un usuario registrado (véase Figura 4.22)
- Escenario 2: Corresponde a las funcionalidades que puede realizar en la plataforma un usuario no registrado (véase Figura 4.23).

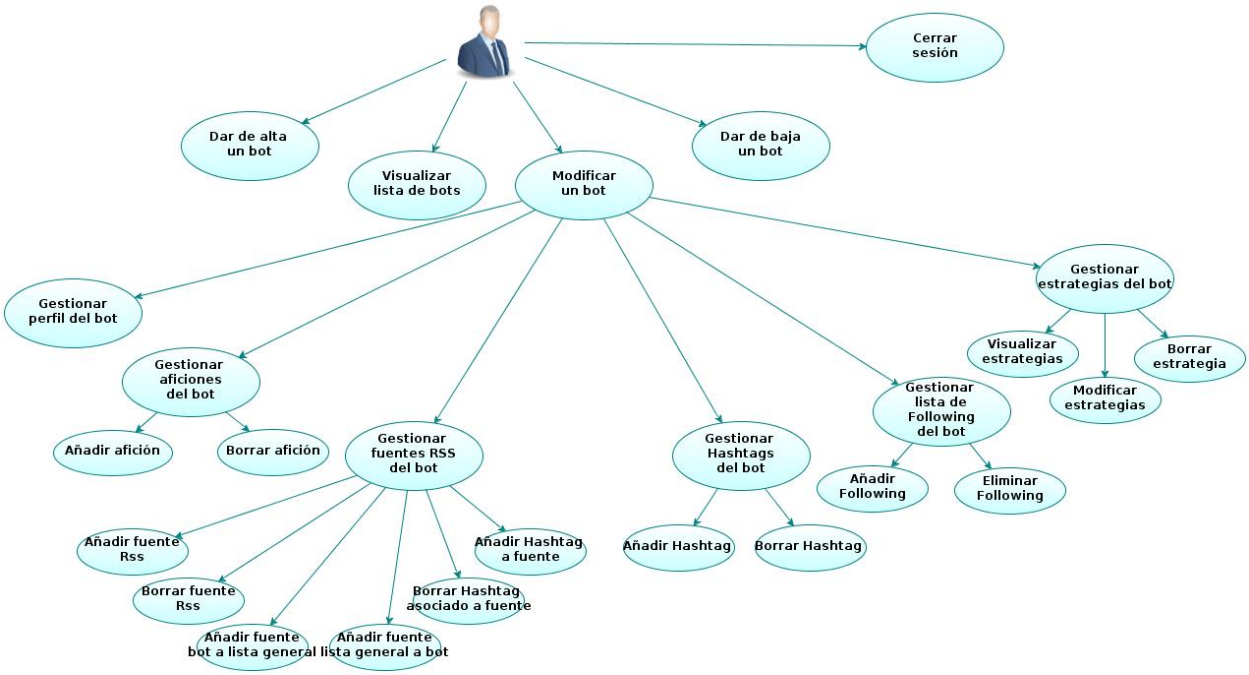


Figura 4.22: Escenario 1: funcionalidades usuario registrado

Figura 4.23: Escenario 2: funcionalidades usuario no registrado

Capítulo 5

Diseño de la aplicación Web TweetyBots

5.1. Introducción

A continuación se realizará una descripción detallada del proceso de diseño de la aplicación Web que utilizará el usuario para crear nuevos bots y configurar su comportamiento.

En primer lugar, se detallarán las funcionalidades que implementará la aplicación para cumplir los requisitos funcionales de la misma, definidos previamente por el usuario 4.4.

A continuación se identificarán las distintas entidades necesarias para el correcto almacenamiento de la información, así como la representación del modelo de datos utilizado.

Una vez definido el modelo de datos, se describirá la lógica de negocio aplicada, así como la representación del diseño en *vistas* de la aplicación.

Por último se realizará una descripción general de la interfaz de usuario, mostrando el diseño estructurado que se ha adoptado durante el diseño de la interfaz gráfica de la aplicación Web.

5.2. Funcionalidades

En este apartado se describen las funcionalidades que implementará la aplicación de gestión y configuración de bots.

- **DAR DE ALTA A USUARIO**: Se permitirá dar de alta a nuevos usuarios en la aplicación. Para ello se pedirá a los nuevos usuarios que indiquen un nombre de usuario y una contraseña. Tras la comprobación de los datos proporcionados por el usuario, se almacenará dicha información en base de datos para el control de autenticación de usuarios. Relacionado con el requisito 4.4.1.

- **INICIAR SESIÓN**: Se permitirá a los usuarios registrados en la aplicación el acceso a la misma, mediante un sistema de autenticación y gestión de usuarios. Relacionado con el requisito 4.5.1.

- **CERRAR SESIÓN**: Se permitirá a los usuarios registrados en la aplicación y que hayan iniciado sesión en la misma, la finalización de la sesión y salida de la aplicación. Relacionado con el requisito 4.5.1.

A continuación se describen el resto de las funcionalidades que proporciona la aplicación a los distintos usuarios registrados en la misma.

- **REGISTRAR UN NUEVO BOT**: Se permitirá registrar un nuevo bot en la aplicación, para que dicha aplicación se encarge de su control y gestión en Twitter. Para ello, es indispensable que la cuenta de Twitter asociada al bot se encuentre previamenta creada. Relacionado con el requisito 4.4.2.

- **DAR DE BAJA UN BOT**: Se permitirá dar de baja un bot dado de alta previamente en la aplicación. Esto no implica dar de baja la cuenta en Twitter asociada al bot, únicamente implica que la aplicación dejará de encargarse de su control y gestión en Twitter, y con ello se borrarán todos los registros asociados al bot que se encuentren en base de datos.Relacionado con el requisito 4.4.2.

- **CONSULTAR LISTA DE BOTS**: Se permitirá consultar los distintos bots dados de alta en la aplicación por un determinado usuario. Relacionado con el requisito 4.4.2.

- **MODIFICAR UN BOT**: Se permitirá modificar el comportamiento de un bot registrado en la aplicación, selecionándolo previamente de la lista de bots registrados por un determinado usuario. Para modificar el comportamiento de un bot, se permitirá modificar la configuración que lo define. Relacionado con el requisito 4.4.2.

  A continuación se detallan las funcionalidades asociadas a la configuración del comportamiento de los distintos bots:

  - **CONSULTAR EL PERFIL DE UN BOT**: Se permitirá consultar el perfil de la cuenta de Twitter asociada al bot. Es decir, se permitirá la consulta de la imagen del perfil, del nombre del bot en Twitter, de la ubicación, de la página Web personal (si la tuviera) y de la biografía del bot. Relacionado con el requisito 4.4.3.

  - **MODIFICAR EL PERFIL DE UN BOT**: Se permitirá modificar el perfil de la cuenta de Twitter asociada al bot. Es decir, se permitirá la modificación de la imagen del perfil, del nombre del bot en Twitter, de la ubicación, de la página Web personal (si la tuviera) y de la biografía del bot. Dicha información quedará registrada en la base de datos. Relacionado con el requisito 4.4.3.

- **AÑADIR AFICIONES A UN BOT**: Se permitirá añadir aficiones a un bot, que proporcionarán información a la aplicación sobre los distintos gustos de dicho bot. Las aficiones que se pueden añadir son: *BRICOLAJE*, *CINE*, *COCINA*, *DEPORTES*, *IDIOMAS*, *JUEGOS*, *LIBROS*, *MODA*, *MOTOR*, *MUSEOS*, *MÚSICA*, *TEATRO*, *TECNOLOGÍA* y *VIAJES*. Dicha información quedará registrada en la base de datos. Relacionado con el requisito 4.4.4.

- **ELIMINAR AFICIÓN ASOCIADA A UN BOT**: Se permitirá eliminar de la base de datos cualquier afición asociada previamente a un bot. Relacionado con el requisito 4.4.4.

- **AÑADIR FUENTES RSS A UN BOT**: Se permitirá añadir fuentes rss a un bot, para que dicho bot pueda transmitir información actual e interesante en Twitter a sus seguidores (***Followers***). Las fuentes rss a añadir se podrán clasificar en las siguientes categorías: *BLOG*, *CIENCIA*, *CINE*, *CULTURA*, *DEPORTES*, *ECOLOGÍA*, *GADGETS*, *IDIOMAS*, *LIBROS*, *MOTOR*, *MUSICA*, *NOTICIAS*, *POLITICA*, *RESTAURANTES*, *TECNOLOGIA*, *TENDENCIAS*, *VIAJES*,*VIDEOJUEGOS* y *VISUALIZACION*. Dicha información quedará registrada en la base de datos. Relacionado con el requisito 4.4.5.

- **ELIMINAR FUENTE RSS ASOCIADA A UN BOT**: Se permitirá borrar de la base de datos una fuentes rss asociada a un bot. Por lo tanto el bot dejará de transmitir en Twitter información relacionada con dicha fuente.

- **VISUALIZAR FUENTES RSS RECOMENDADAS**: Se permitirá visualizar el listado público y común de fuentes Rss generales recomendadas por la aplicación. Relacionado con el requisito 4.4.5.

- **AÑADIR FUENTE RSS RECOMENDADA A UN BOT**: Se permitirá añadir cualquier fuente disponible en el listado público y común de fuentes Rss generales recomendadas por la aplicación a un bot determinado. Relacionado con el requisito 4.4.5.

- **AÑADIR NUEVA FUENTE A LISTA DE FUENTES RSS RECOMENDADAS**: Se permitirá añadir una nueva fuente al listado público y común de fuentes Rss generales recomendadas por la aplicación. Para ello deberá seleccionarse previamente la fuente a añadir del listado de fuentes rss del bot. Relacionado con el requisito 4.4.5.

- **AÑADIR HASHTAG A UN BOT**: Se permitirá añadir un nuevo ***Hashtag***, para que el bot pueda buscar información interesante relacionada con dicho hashtag en Twitter y poder retransmitir dicha información a sus seguidores. Los hashtags podrán añadirse de uno en uno, o bien en forma de lista separados por comas (Ej.: #f1, #ferrari). Los hashtags asociados a un determinado bot

quedarán registrados en la base de datos. Relacionado con el requisito 4.4.6.

- **ELIMINAR HASHTAG ASOCIADO A UN BOT**: Se permitirá borrar de la base de datos un hashtag asociado a un bot. Por lo tanto el bot dejará de buscar en Twitter información relacionada con dicho hashtag. Relacionado con el requisito 4.4.6.

- **AÑADIR NUEVO FOLLOWING A UN BOT**: Se permitirá añadir un nuevo ***Followings*** a un bot, esto es, un nuevo usuario para que el bot comience a seguir en Twitter. De esta forma el bot podrá leer el ***Timeline*** del following añadido y retransmitir la información que considere importante al resto de los usuarios que siguen a dicho bot en Twitter. Los usuarios seguidos por un determinado bot quedarán registrados en la base de datos. Relacionado con el requisito 4.4.7.

- **ELIMINAR FOLLOWING ASOCIADO A UN BOT**: Se permitirá eliminar un following asociado a un determinado bot. Por lo tanto el bot dejará de seguir a dicho usuario en Twitter. Relacionado con el requisito 4.4.7.

- **VISUALIZAR ESTRATEGIAS DE COMPORTAMIENTO DE UN BOT**: Se permitirá visualizar las estrategias de comportamiento asociadas a un determinado bot. Relacionado con el requisito 4.4.8:
  1. **Visualizar estrategia de following de un bot**: Se permitirá visualizar la estrategia de following establecida para un determinado bot, esto es, el modo en que el bot sigue a otros usuarios de Twitter.

  2. **Visualizar estrategia de cortesía de un bot**: Se permitirá visualizar la estrategia de cortesía establecida para un determinado bot, esto es, el modo en que el bot agradece los cumplidos que le hayan realizado otros usuarios de Twitter.

  3. **Visualizar estrategia de personalidad de un bot**: Se permitirá visualizar la estrategia de personalidad establecida para un determinado bot, esto es, el tipo de personalidad que el bot desarrollará a la hora de comunicarse con otros usuarios en Twitter.

  4. **Visualizar estrategia de Follow on Friday de un bot**: Se permitirá visualizar la estrategia de ***Follow on Friday*** establecida para un determinado bot.

  5. **Visualizar estrategia de publicación de un bot**: Se permitirá visualizar la estrategia de publicación establecida para un determinado bot, esto es, la frecuencia con la que el bot realizará publicaciones en Twitter.

- **MODIFICAR ESTRATEGIAS DE COMPORTAMIENTO DE UN BOT**: Se permitirá modificar las estrategias de comportamiento asociadas a un determinado bot. Relacionado con el requisito 4.4.8:

  1. **Modificar estrategia de following de un bot**: Se permitirá modificar la estrategia de following establecida para un determinado bot. Para ello el usuario podrá modificar:
     - El número máximo de followings que realizará el bot al día.
     - El perfil que han de cumplir los usuarios a los que el bot seguirá en Twitter (perfil de following)
     - Las condiciones que se han de cumplir para que el bot comience a seguir a un usuario en Twitter.
     - Las condiciones que se han de cumplir para que el bot deje de seguir a un usuario en Twitter.

     La configuración de dicha estrategia quedará almacenada en base de datos.

  2. **Modificar estrategia de cortesía de un bot**: Se permitirá modificar la estrategia de cortesía establecida para un determinado bot. Para ello el usuario podrá:
     - Indicar si el bot enviará o no, mensajes de agradecimiento por cada ***ReTweet*** realizado por otro usuario en Twitter, así como modificar la forma en la que se enviarán dichos agradecimientos: a través de mensaje directo (DM) o bien través de su ***Timeline*** público.
     - Indicar si el bot enviará o no, mensajes de agradecimiento por cada Follow on Friday (#FF) realizados por otros usuarios en Twitter a dicho bot, a través de su ***Timeline***, así como indicar si el bot corresponderá igualmente a estos usuarios realizando un Follow on Friday (#FF) de vuelta.
     - Indicar si el bot enviará o no, mensajes de agradecimiento a nuevos ***Followers*** a través de mensajes directos (DM).

     La configuración de dicha estrategia quedará almacenada en base de datos.

  3. **Modificar estrategia de personalidad de un bot**: Se permitirá modificar la estrategia de personalidad establecida para un determinado bot. Para ello el usuario podrá:
     - Indicar si el bot dará o no los 'Buenos días'a sus seguidores en Twitter, así como modificar la forma en la que se darán los 'Buenos días': de forma general o personalizada a través de su **timeline** público.
     - Indicar si el bot dará o no las 'Buenas noches'a sus seguidores en Twitter de forma generalizada a través de su **timeline** público.
     - Indicar si el bot informará o no sobre lo que está comiendo a sus seguidores en Twitter a través de su **timeline** público.
     - Indicar si el bot informará de actividades que realiza con su familia a sus seguidores en Twitter a través de su **timeline** público.

- Indicar si el bot enviará mensajes de tipo ***TGIF (Thanks God It's Friday)*** los viernes a través de su ***Timeline*** público.

La configuración de dicha estrategia quedará almacenada en base de datos.

4. **Modificar estrategia de Follow on Friday de un bot**: Se permitirá modificar la estrategia de ***Follow on Friday*** establecida para un determinado bot. Para ello el usuario podrá modificar:
   - El número máximo de mensajes de tipo #FF que realizará el bot los viernes.
   - El número máximo de usuarios que se incluirán por cada mensaje de tipo #FF.
   - El perfil que han de cumplir los usuarios a los que el bot mencionará en mensajes de tipo #FF los viernes.

   La configuración de dicha estrategia quedará almacenada en base de datos.

5. **Modificar estrategia de publicación de un bot**: Se permitirá modificar la estrategia de publicación establecida para un determinado bot. Para ello el usuario podrá modificar:
   - El número máximo de ***Tweets*** que el bot enviará a lo largo del día.
   - El número máximo de ***ReTweets*** que el bot enviará a lo largo del día.
   - El horario de publicación del bot en los días laborables (L-V).
   - El horario de publicación del bot en los días festivos (S-D).
   - Las horas entre las que el bot podrá informar a sus usuarios de lo que se encuentra comiendo (si implementa la estrategia de personalidad).
   - La hora en la que el bot enviará mensajes con mayor frecuencia (hora punta).

   La configuración de dicha estrategia quedará almacenada en base de datos.

5.3. Diseño del modelo de datos

Para el diseño de la capa de persistencia de la aplicación Web, se han tenido en cuenta las entidades lógicas necesarias para el correcto almacenamiento de la información.

En la Figura 5.1, se muestra el esquema relacional que describe con detalle el diseño de la base de datos de la aplicación Web:

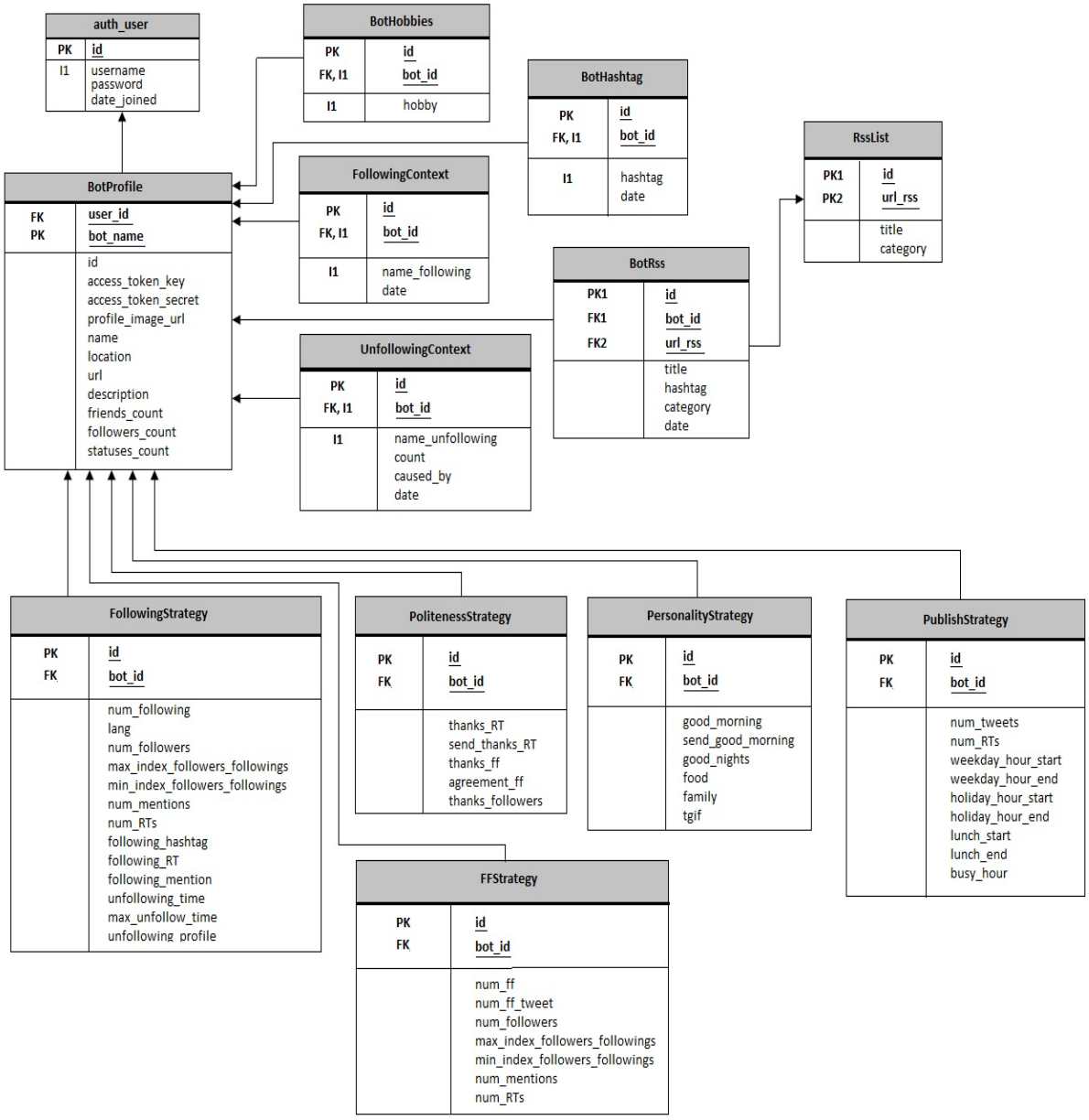


Figura 5.1: Diagrama relacional del modelo de datos

A continuación se describen las principales entidades lógicas de la aplicación Web:

- **USUARIO**: Almacena la información asociada a los usuarios registrados en la aplicación. Un usuario vendrá definido por un nombre de usuario (*login*) que lo identifica unívocamente en la base de datos y una contraseña.

- **PERFIL DEL BOT**: Almacena la información asociada a un bot. Un bot vendrá definido por su nombre (por el que se identifica unívocamente en la base de datos y en Twitter) junto con el identificador en Twitter del bot (por el que se identifica a un usuario en Twitter), un par de claves que permitirán el acceso de la aplicación a la cuenta en Twitter del bot (*access_token_key* y *access_token_secret*), la información del perfil de Twitter del bot: la dirección Url asociada a la imagen del perfil, el nombre del bot, su ubicación, la dirección Url de la página Web personal (si la tuviera) y su biografía; el número de followings del bot, el número de followers y el número de Tweets publicados por el bot.

- **AFICIONES DEL BOT**: Almacena la información correspondiente a las aficiones del bot, que vendrá dada por un identificador con el cual se identifica unívocamente en la base de datos y una categoría de afición. Las distintas categorías de aficiones contempladas son: *BRICOLAJE*, *CINE*, *COCINA*, *DEPORTES*, *IDIOMAS*, *JUEGOS*, *LIBROS*, *MODA*, *MOTOR*, *MUSEOS*, *MÚSICA*, *TEATRO*, *TECNOLOGÍA* y *VIAJES*.

- **FUENTES RSS DEL BOT**: Almacena la información correspondiente a las fuentes Rss (*feed Rss*) del bot, que vendrá dada por un identificador con el cual se identifica unívocamente en la base de datos, la dirección Url, el título y la categoría de la fuente, así como un hashtag asociado a dicha fuente que será de caracter no obligatorio, y la fecha en la que se almacenó la fuente en la base de datos. Los distintos tipos de categorías en los que se pueden clasificar las fuentes Rss del bot son: *BLOG*, *CIENCIA*, *CINE*, *CULTURA*, *DEPORTES*, *ECOLOGIA*, *GADGETS*, *IDIOMAS*, *LIBROS*, *MOTOR*, *MUSICA*, *NOTICIAS*, *POLITICA*, *RESTAURANTES*, *TECNOLOGIA*, *TENDENCIAS*, *VIAJES*, *VIDEOJUEGOS* y *VISUALIZACION*.

- **LISTA DE FUENTES GENERALES**: Almacena la información correspondiente a la lista de fuentes Rss (*feed Rss*) generales de la aplicación, que vendrá dada por un identificador con el cual se identifica unívocamente en la base de datos, junto con la dirección Url, el título y la categoría de la fuente. Los distintos tipos de categorías en los que se pueden clasificar las fuentes del bot son: *BLOG*, *CIENCIA*, *CINE*, *CULTURA*, *DEPORTES* , *ECOLOGIA*, *GADGETS*, *IDIOMAS*, *LIBROS*, *MOTOR*, *MUSICA*, *NOTICIAS*, *POLITICA*, *RESTAURANTES*, *TECNOLOGIA*, *TENDENCIAS*, *VIAJES*, *VIDEOJUEGOS* y *VISUALIZACION*.

- **HASHTAGS DEL BOT**: Almacena la información correspondiente a los hashtags favoritos del bot, que vendrá dada por un identificador con el cual se identifica unívocamente en la base de datos, el valor del hashtag y la fecha en la que se añadió dicho hashtag al bot.

- **CONTEXTO DE FOLLOWING**: Almacena la información correspondiente a los usuarios a los que el bot se encuentra siguiendo actualmente en Twitter, que vendrá dada por un identificador con el cual se identifica unívocamente en la base de datos, el nombre del usuario al que se encuentra siguiendo el bot en Twitter (*following*) y la fecha en la que el bot comenzó a seguir a dicho usuario.

- **CONTEXTO DE UNFOLLOWING**: Almacena la información correspondiente a los usuarios a los que el bot ha dejado de seguir en Twitter, que vendrá dada por un identificador con el cual se identifica unívocamente en la base de datos, el nombre del usuario al que se ha dejado de seguir en Twitter, el motivo por el que se ha dejado de seguir al usuario, el número de veces que se le ha dejado de seguir y la fecha en la que se ha dejado de seguir a dicho usuario en Twitter. Los motivos por los que un bot podrá dejar de seguir a un usuario en Twitter son: porque el usuario ha dejado de cumplir el perfil de following (si el bot implementa la estrategia), porque el usuario no ha correspondido al following del bot en un plazo determinado (si el bot implementa la estrategia) o bien porque se ha dejado de seguir a dicho usuario a través de la aplicación de *TweetyBots*.

- **ESTRATEGIA DE FOLLOWING**: Almacena la información correspondiente a la configuración de la estrategia de following establecida para cada bot, que vendrá dada por el número máximo de followings que realizará el bot al día, el perfil de following: el lenguaje en el que escribirán los nuevos followings, el mínimo número de followers que deberá tener dicho following, el índice de followers/followings máximo y mínimo, el número mínimo de menciones y de retweets (RTs) que se le deberán de haber realizado al nuevo following en los últimos 7 días; las condiciones que se han de cumplir para que el bot comience a seguir a un usuario en Twitter: por uso de hashtags favoritos, por realizar retweets (RTs) a nuestro bot y/o por realizar menciones a nuestro bot; y las condiciones que se han de cumplir para que el bot deje de seguir a un usuario en Twitter: por no corresponder en un tiempo máximo y/o por dejar de cumplir el perfil de following.

- **ESTRATEGIA DE CORTESÍA**: Almacena la información correspondiente a la configuración de la estrategia de cortesía establecida para cada bot, que vendrá dada por un booleano que indique si el bot enviará o no, mensajes de agradecimiento por cada retweet (RT) realizado por otro usuario en Twitter al bot, así como la forma en la que se enviarán dichos agradecimientos: a través de mensaje directo (DM) o bien través de su timeline público; un booleano que indique si el bot enviará o no, mensajes de agradecimiento a través de su timeline público por cada

Follow on Friday (#FF) realizado por otro usuario en Twitter al bot, así como un booleano que indique si el bot corresponderá igualmente a estos usuarios realizando un Follow on Friday (#FF) de vuelta; y un booleano que indique si el bot enviará o no, mensajes de agradecimiento a nuevos followers a través de mensajes directos (DM).

- **ESTRATEGIA DE PERSONALIDAD**: Almacena la información correspondiente a la configuración de la estrategia de personalidad establecida para cada bot, que vendrá dada por un booleano que indique si el bot dará o no los "Buenos días" a sus seguidores en Twitter, así como la forma en la que se darán los "Buenos días": de forma general o personalizada a través de su timeline público; un booleano que indique si el bot dará o no las "Buenas noches" a sus seguidores en Twitter de forma generalizada a través de su timeline público, un booleano que indique si el bot informará o no a sus seguidores en Twitter sobre lo que está comiendo a través de su timeline público, un booleano que indique si el bot informará a sus seguidores en Twitter de actividades que realiza con su familia a través de su timeline público; y un booleano que indique si el bot enviará mensajes de tipo TGIF (Thanks God is Friday) los viernes a través de su timeline público.

- **ESTRATEGIA DE FOLLOW ON FRIDAY**:Almacena la información correspondiente a la configuración de la estrategia de Follow on Friday (#FF) establecida para cada bot, que vendrá dada por el número máximo de mensajes de tipo Follow on Friday (#FF) que realizará el bot los viernes, el número máximo de usuarios que se incluirán por cada mensaje de tipo Follow on Friday (#FF) y el perfil que han de cumplir los usuarios a los que el bot realizará Follow on Friday los viernes: el número mínimo de followers que ha de tener el usuario, el índice de followers/followings máximo y mínimo que ha de cumplir el usuario, así como el número mínimo de menciones y de retweets (RTs) realizados a dicho usuario en los últimos 7 días.

- **ESTRATEGIA DE PUBLICACIÓN**: Almacena la información correspondiente a la configuración de la estrategia de publicación establecida para cada bot, que vendrá dada por el número máximo de tweets y el número máximo de retweets (RTs) que el bot enviará a lo largo del día, el horario de publicación del bot en los días laborables (L-V), el horario de publicación del bot en los días festivos (S-D), las horas entre las que el bot podrá informar a sus usuarios de lo que se encuentra comiendo (si implementa la estrategia de personalidad) y la hora en la que el bot enviará mensajes con mayor frecuencia (hora punta).

5.4. Diseño de la lógica de negocio de la aplicación

Puesto que la capa lógica de negocio de la aplicación viene dada por el patrón de diseño MVC definido por el framework utilizado (*Django*), en este apartado se hablará de las *vistas* que componen la aplicación, puesto que son las encargadas de gestionar la lógica de negocio de la misma.

Una *vista* es una simple función de Python que toma como argumento una petición Web y devuelve una respuesta Web, incluyendo en sí misma toda la lógica necesaria para retornar dicha respuesta.

En la Figura [5.2] y en la Figura [5.3] se muestra el diagrama de vistas de la aplicación en el que se muestra gráficamente la iteracción entre vistas, así como la iteracción entre cada vista con la base de datos correspondiente.

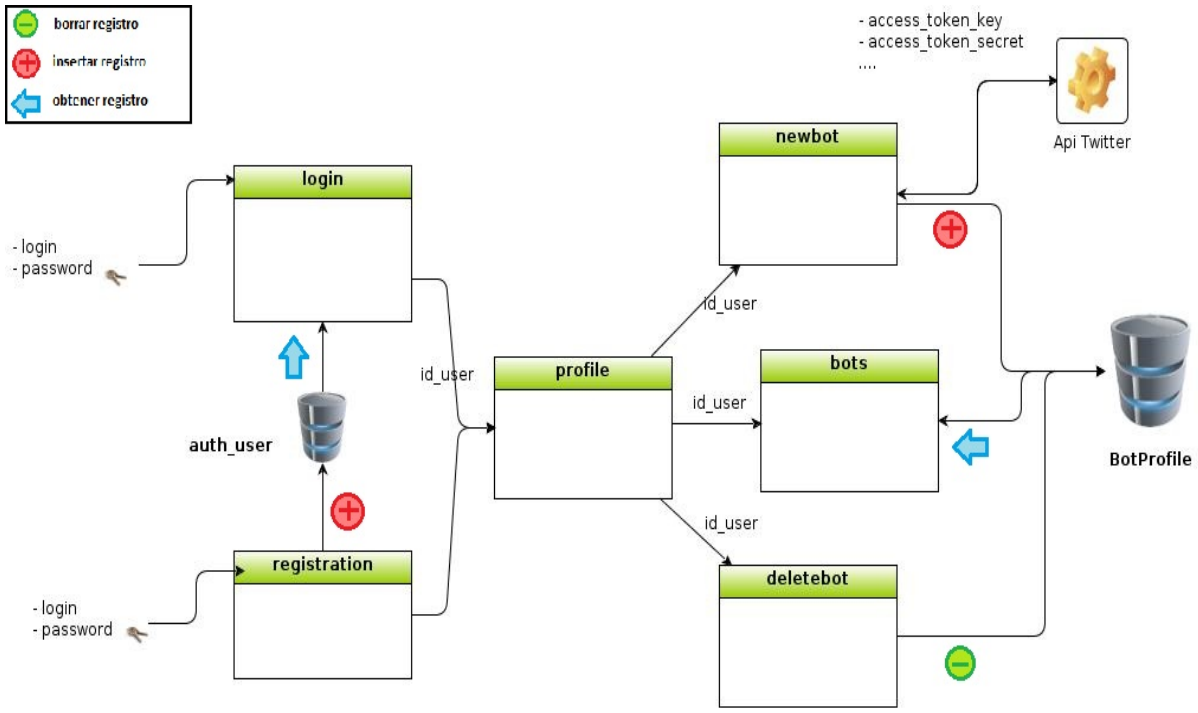


Figura 5.2: Diagrama de vistas(I)

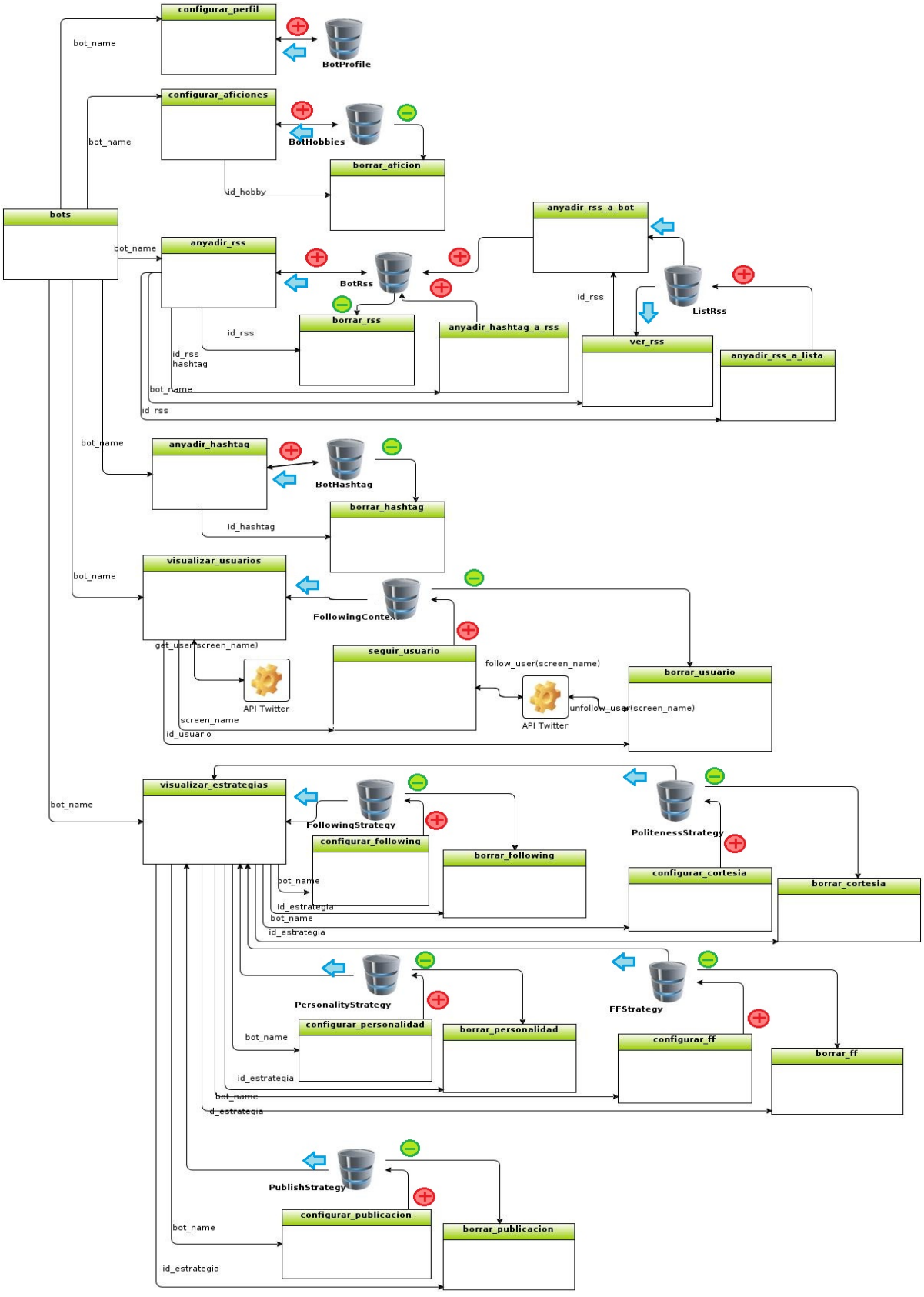


Figura 5.3: Diagrama de vistas(II)

Tras el estudio de los requisitos funcionales de la aplicación 4.4, las principales *vistas* que se han obtenido en la fase de diseño y que componen la capa lógica de la aplicación son:

- **registration(request)**: Vista que gestiona el registro de un usuario. Dicha vista recibe los datos del registro de un usuario a través de un formulario, y tras validar que los datos recibidos son correctos y comprobar que no existe un usuario registrado en la aplicación con el mismo *login* introducido, se encarga de almacenar los datos del usuario en la base de datos correspondiente (`Tabla auth_user` [5.3]) proporcionándole de este modo el acceso a la aplicación.

- **profile(request)**: Vista que gestiona la visualización de la página inicial que se muestra al usuario registrado. Dicha vista se encargará de gestionar el menú de opciones que permitirá a un usuario: *Crear un nuevo bot*, *Visualizar sus bots* y *Borrar un bot* dado de alta en la aplicación.

- **newbot(request)**: Vista que gestiona el proceso de dar de alta a un nuevo bot en la aplicación. Dicha vista se encarga de realizar las llamadas necesarias al API de Twitter para la obtención de las claves del usuario (*access_token_key* y *access_token_secret*) que permitirán a la aplicación interactuar con Twitter por mediación del usuario de forma segura. Una vez obtenidas las claves se almacenarán en la base de datos correspondiente junto con algunos datos en Twitter del usuario: *screen_name*, *id*, *profile_image_url*, *name*, *location*, *url*, *description*, *friends_count*, *followers_count* y *statuses_count* (`Tabla BotProfile` [5.3]).

- **bots(request)**: Vista que gestiona la visualización de los distintos bots dados de alta por un usuario registrado en la aplicación. Para ello deberá consultar dicha información de la base de datos correspondiente (`Tabla BotProfile` [5.3]).

- **configurar_perfil(request, bot_name)**: Vista que gestiona la visualización/modificación del perfil en Twitter del bot. Dicha vista se encarga de realizar las llamadas necesarias al API de Twitter, para la obtención de los datos del perfil en Twitter del bot, para mostrarlos posteriormente al usuario, así como para la modificación por parte del usuario de dichos datos. Una vez obtenidos los datos o bien una vez realizados los cambios pertinentes, se almacenará la información obtenida en la base de datos correspondiente (`Tabla BotProfile` [5.3]).

- **configurar_aficiones(request, bot_name)**: Vista que gestiona la adición de una nueva afición a la lista de aficiones de un bot. Dicha vista recibe la información de la nueva afición a través de un formulario y tras validar que la información recibida es correcta y comprobar que no existe dicha afición en la lista de aficiones del bot, se encarga de almacenar los datos de la nueva afición en la base de datos correspondiente (`Tabla BotHobbies` [5.3]). Así mismo, dicha vista se encarga de la visualización en

una tabla de datos de la lista de aficiones del bot.

- **borrar_aficion(request, bot_name, id)**: Vista que gestiona la eliminación de una afición de la lista de aficiones de un bot. Dicha vista recibe como parámetro el identificador de la afición, y tras comprobar que los datos proporcionados son correctos, se encarga de eliminar los datos de la afición de la base de datos correspondiente (`Tabla BotHobbies` [5.3]).

- **anyadir_rss(request, bot_name)**: Vista que gestiona la adición de una nueva fuente Rss a la lista de fuentes Rss de un bot. Dicha vista recibe la información de la nueva fuente a través de un formulario: *Url de la fuente Rss* y *categoría*; y tras validar que los datos recibidos son correctos y comprobar que no existe dicha fuente en la lista de fuentes Rss del bot, se encarga de almacenar los datos de la nueva fuente en la base de datos correspondiente (`Tabla BotRss` [5.3]). Así mismo, dicha vista se encarga de la visualización en una tabla de datos de la lista de fuentes Rss del bot.

- **borrar_rss(request, bot_name, id)**: Vista que gestiona la eliminación de una fuente Rss de la lista de fuentes Rss de un bot. Dicha vista recibe como parámetro el identificador de la fuente, y tras comprobar que los datos proporcionados son correctos, se encarga de eliminar los datos de la fuente de la base de datos correspondiente (`Tabla BotRss` [5.3]).

- **anyadir_rss_a_lista(request, bot_name, id)**: Vista que gestiona la adición de una fuente Rss de la lista de fuentes Rss de un bot, a la lista general de fuentes Rss de la aplicación. Dicha vista recibe como parámetro el identificador de la fuente Rss del bot, y tras comprobar que los datos proporcionados son correctos, se encarga de añadir dicha fuente a la lista general de fuentes de la aplicación, almacenando los datos de la nueva fuente en la base de datos correspondiente (`Tabla RssList` [5.3]).

- **anyadir_rss_a_bot(request, bot_name, id)**: Vista que gestiona la adición de una fuente Rss de la lista general de fuentes de la aplicación a la lista de fuentes Rss de un bot. Dicha vista recibe como parámetro el identificador de la fuente Rss de la lista general de fuentes de la aplicación, y tras comprobar que los datos proporcionados son correctos, se encarga de añadir dicha fuente a la lista de fuentes Rss del bot, almacenando los datos de la nueva fuente en la base de datos correspondiente (`Tabla BotRss` [5.3]).

- **ver_rss(request, bot_name)**: Vista que gestiona la visualización por parte del usuario de la lista general de fuentes de la aplicación. Dicha vista consulta la información disponible en la base de datos correspondiente (`Tabla ListRss` [5.3]) y se encarga de mostrarla en una tabla de datos.

- **anyadir_hashtag_a_rss(request, bot_name, id, hashtag)**: Vista que gestiona la asociación de un hashtag a una fuente Rss de la lista de fuentes Rss de un bot. Dicha vista recibe como parámetro el identificador de la fuente Rss a la que se quiere asociar el hashtag y el hashtag (o la lista de hashtags separados por coma), y tras comprobar que los datos proporcionados son correctos, se encarga de asociar dicho hashtag a la fuente Rss indicada, almacenando los datos del nuevo hashtag en la base de datos correspondiente (`Tabla BotRss` [5.3]).

- **anyadir_hashtags(request, bot_name)**: Vista que gestiona la adición de un nuevo hashtag a la lista de hashtags favoritos de un bot. Dicha vista recibe la información del nuevo hashtag a través de un formulario y tras validar que los datos recibidos son correctos y comprobar que no existe dicho hashtag en la lista de hashtags favoritos del bot, se encarga de almacenar los datos del nuevo hashtag en la base de datos correspondiente (`Tabla BotHashtag` [5.3]). Así mismo, dicha vista se encarga de la visualización en una tabla de datos de la lista de hashtags favoritos del bot.

- **borrar_hashtag(request, bot_name, id)**: Vista que gestiona la eliminación de un hashtag de la lista de hashtags favoritos de un bot. Dicha vista recibe como parámetro el identificador del hashtag, y tras comprobar que los datos proporcionados son correctos, se encarga de eliminar los datos del hashtag de la base de datos correspondiente (`Tabla BotHashtag` [5.3]).

- **visualizar_usuarios(request, bot_name)** Vista que gestiona la visualización por parte del usuario de la lista de ***Following*** de un bot. Dicha vista consulta la información disponible en la base de datos correspondiente (`Tabla FollowingContext` [5.3]) y se encarga de mostrarla en una tabla de datos. Así mismo, dispone al usuario de un mecanismo para añadir nuevos followings al bot. Dicha vista recibe a través de un formulario el nombre de un usuario de Twitter (Ej.: pepito) y se encarga de realizar las llamadas necesarias al API de Twitter para la obtención de la información del perfil de Twitter de dicho usuario.

- **seguir_usuario(request, bot_name, screen_name)**: Vista que gestiona la adición de un nuevo ***Following*** a la lista de followings de un bot. Dicha vista recibe como parámetro el nombre en Twitter del usuario que se quiere comenzar a seguir, y tras comprobar que los datos proporcionados son correctos, se encarga de realizar las llamadas necesarias al API de Twitter para que un determinado bot comience a seguir en Twitter al usuario indicado. Una vez realizada con éxito la operación, se almacenará la información necesaria en la base de datos correspondiente (`Tabla FollowingContext` [5.3]).

- **borrar_usuario(request, bot_name, id)**: Vista que gestiona la eliminación de un ***Following*** de la lista de followings de un bot. Dicha vista recibe como parámetro

el identificador del following, y tras comprobar que los datos proporcionados son correctos, se encarga de realizar las llamadas necesarias al API de Twitter para que un determinado bot deje de seguir en Twitter al usuario indicado. Una vez realizada con éxito la operación, se eliminarán los datos del following de la base de datos correspondiente (`Tabla FollowingContext` [5.3]).

- **visualizar_estrategias(request, bot_name)**: Vista que gestiona la visualización por parte del usuario de las estrategias disponibles en la aplicación para configurar el comportamiento de un bot. Dicha vista consulta la información disponible en las bases de datos correspondientes (`Tabla FollowingStrategy`, `Tabla PolitenessStrategy`, `Tabla PersonalityStrategy`, `Tabla FFStrategy` y `Tabla PublishStrategy` [5.3]) y se encarga de mostrar dicha información en un menú de datos desplegable.

- **configurar_following(request, bot_name)**: Vista que se gestiona la configuración de la estrategia de following de un bot. Dicha vista recibe la información de la nueva configuración de la estrategia a través de un formulario y tras validar que los datos recibidos son correctos, se encarga de almacenar la nueva configuración en la base de datos correspondiente (`Tabla FollowingStrategy` [5.3]).

- **configurar_cortesia(request, bot_name)**: Vista que se gestiona la configuración de la estrategia de cortesía de un bot. Dicha vista recibe la información de la nueva configuración de la estrategia a través de un formulario y tras validar que los datos recibidos son correctos, se encarga de almacenar la nueva configuración en la base de datos correspondiente (`Tabla PolitenessStrategy` [5.3]).

- **configurar_personalidad(request, bot_name)**: Vista que se gestiona la configuración de la estrategia de personalidad de un bot. Dicha vista recibe la información de la nueva configuración de la estrategia a través de un formulario y tras validar que los datos recibidos son correctos, se encarga de almacenar la nueva configuración en la base de datos correspondiente (`Tabla PersonalityStrategy` [5.3]).

- **configurar_ff(request, bot_name)**: Vista que se gestiona la configuración de la estrategia de ***Follow on Friday*** (#FF) de un bot. Dicha vista recibe la información de la nueva configuración de la estrategia a través de un formulario y tras validar que los datos recibidos son correctos, se encarga de almacenar la nueva configuración en la base de datos correspondiente (`Tabla FFStrategy` [5.3]).

- **configurar_publicacion(request, bot_name)**: Vista que se gestiona la configuración de la estrategia de publicación de un bot. Dicha vista recibe la información de la nueva configuración de la estrategia a través de un formulario y tras validar que los datos recibidos son correctos, se encarga de almacenar la nueva configuración en la

base de datos correspondiente (`Tabla PublishStrategy` [5.3]).

5.5. Diseño de la interfaz de usuario

La interfaz gráfica de la aplicación estará formada por las diversas plantillas HTML que componen la arquitectura MVC empleada. El sistema de plantillas en el que se basa el *framework* utilizado permite separar la parte dinámica de la página Web del código estático. Se tienen, por tanto, un conjunto de páginas Web dinámicas implementadas con HTML y Javascript.

El diseño de la interfaz es uno de los elementos clave de la realización del aplicación Web, puesto que es el medio a través del cual los usuarios pueden comunicarse con el programa. Por ello se ha decidido que la interfaz sea sencilla e intuitiva para facilitar en la medida de lo posible la interacción del usuario con la misma.

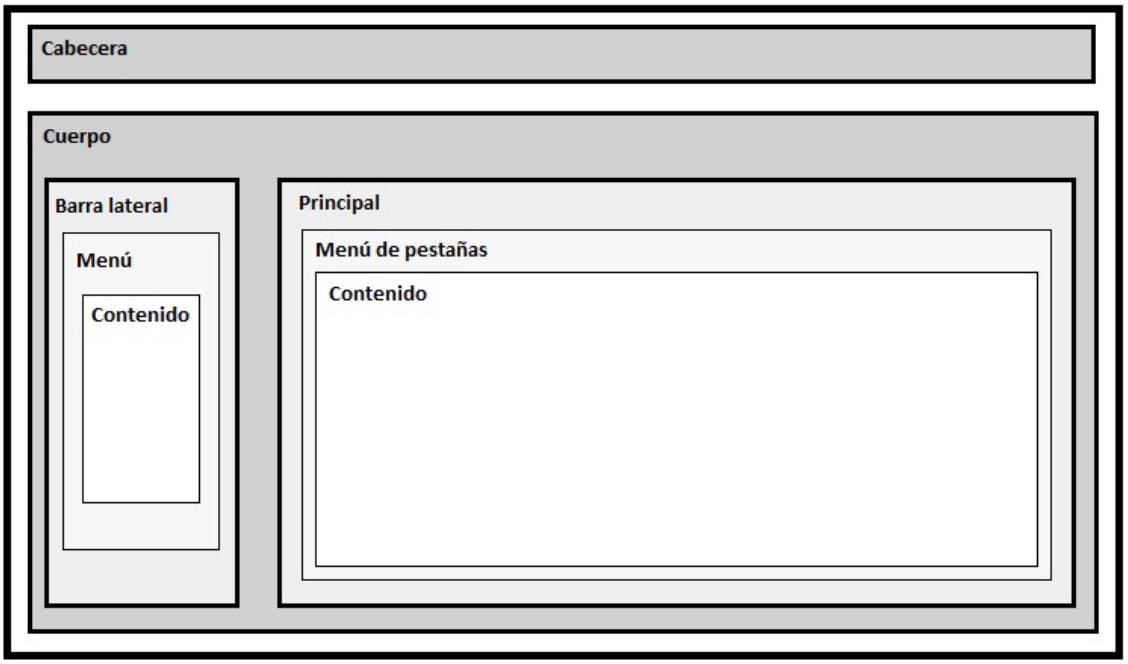


Figura 5.4: Diseño interfaz web

Tal y como se observa en la Figura 5.4, en la que se muestra el diseño que se ha adoptado durante el diseño de la interfaz, se pueden diferenciar claramente dos partes: la cabecera y el cuerpo.

En la cabecera se mostrará siempre el botón de "Cierre de sesión" en todas las vistas de la aplicación Web, para que el usuario pueda finalizar la sesión en cualquier momento durante la navegación, junto con su nombre de usuario.

El cuerpo, a su vez, se divide en dos partes: Por un lado se observa la zona principal, donde se mostrarán las diferentes opciones para que el usuario pueda configurar el comportamiento de un bot. En esta zona, se permitirá navegar entre las diferentes opciones a través de un menú de pestañas sencillo e intuitivo para el usuario. Por otro lado se observa la barra lateral que muestra un menú a través del cual el usuario podrá seleccionar el resto de las funcionalidades que proporciona la aplicación a los distintos usuarios registrados del sistema: *Dar de alta un bot*, *Dar de baja un bot* y *Visualizar bots dados de alta por el usuario*.

Todo el texto contenido en las vistas que forman la aplicación Web estará escrito en castellano o en términos técnicos comprensibles en dicha lengua.

Y por útimo, se debe destacar que la interfaz se encuentra optimizada para Internet Explorer 7.0 o superior y Mozilla Firefox 3 o superior.

Capítulo 6

Diseño del comportamiento de los bots

6.1. Introducción

En este capítulo se detallará el proceso de diseño de las funcionalidades de los distintos Bots. Para el diseño de dichas funcionalidades se han optado por el desarrollo de un conjunto de **procedimientos** parametrizados que se ejecutarán periódicamente y que se encargarán de ejecutar cada una de las tareas que realizará el bot en Twitter. Dichos procedimientos componen el diseño de la lógica de negocio que se encarga de dirigir el comportamiento de los bots en Twitter.

Igualmente en este capítulo se mostrará el diseño del modelo de datos que hace posible el intercambio de información entre los distintos procedimientos, así como el almacenamiento persistente de información necesaria para el correcto funcionamiento de los mismos.

6.2. Procedimientos

A continuación se muestra un cuadro resumen con los procedimientos que se han llevado a cabo para la simulación del comportamiento humano en Twitter.

Por cada procedimiento se especifica: La frecuencia de ejecución, el contexto de la tarea y las acciones en Twitter que conlleva la ejecución de dicha tarea.

Para dotar de periodicidad a los procedimientos se ha decidido incluir cada uno de ellos en una tarea de cron. Por ello la notación que se especifica para la frecuencia es la del cron. Se indican posicionalmente los minutos (0-60), las horas (0-24) y por último el día de la semana (1-7). El valor (*) indica que se repite siempre.

Cuadro 6.1: Resumen procedimientos

Tarea	**Frecuencia**	**Contexto**	**Follow**	**Unfollow**	**RT**	**Tweet**	**DM**
search_RTs	30 8-23 * * *	Agradecimientos enviados	X			X	X
read_TL	25 * * * *	Último id de búsqueda y RTs realizados			X		
search_hashtags	45 8-23 * * *	Último id de búsqueda y RTs realizados	X		X		
read_rss	15 * * * *	Última URL consultada de cada fuente					
write_goodmorning	0 7-11 * * *	Tweets publicados por el bot				X	
write_goodnights	50 22,23,0,1,2 * * *	Tweets publicados por el bot				X	
write_food	55 * * * *	Tweets publicados por el bot				X	
write_family	10 * * * *	Tweets publicados por el bot Contexto familiar Contexto actividades familiares				X	
write_TGIF	0 7-22 * * 5	Tweets publicados por el bot				X	
write_RSS	20 * * * *	Tweets publicados por el bot				X	
write_FF	35 * * * 5	Tweets publicados por el bot y #FF realizados				X	
search_FF		Tweets publicados por el bot, #FF realizados y agradecimientos enviados				X	
check_following	0 0 * * *	Contexto de Unfollowing		X			
check_followers	0 10 * * *	DM de agradecimiento enviados	X				X

Cuadro 6.2: Resumen procedimientos (Cont.)

Tarea	Frecuencia	Contexto	Follow	Unfollow	RT	Tweet	DM
reset_rss	0 0 * * *						
reset_hashtags	0 0 * * *						

6.3. Diseño del modelo de datos

Para el diseño de la capa de persistencia de los procedimientos que se encargan de simular el comportamiento del bot, se han tenido en cuenta las entidades lógicas necesarias para el correcto almacenamiento de la información. Dichas entidades, junto con las ya vistas en el Apartado 5.3 forman el modelo de datos de la plataforma Web.

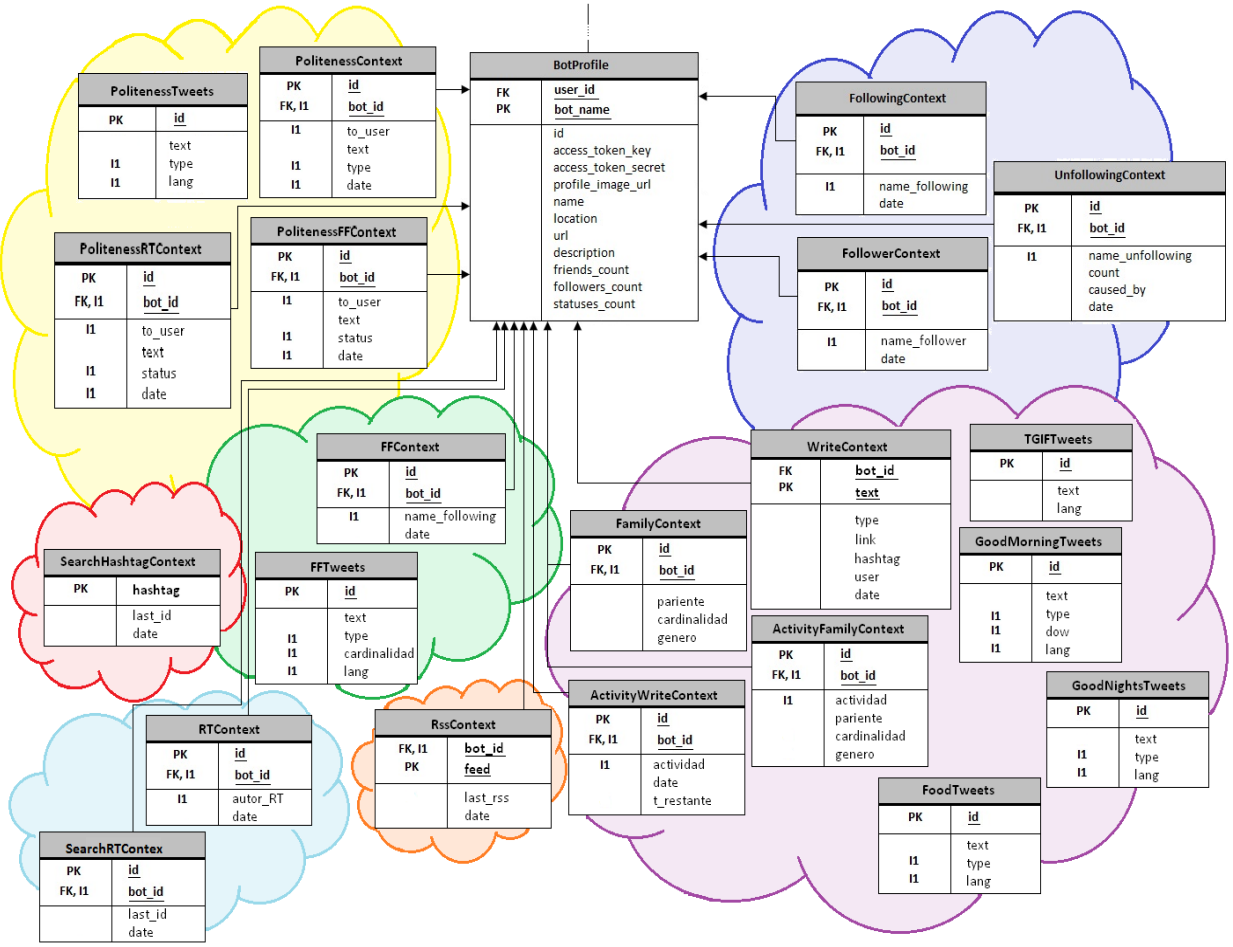


Figura 6.1: Modelo de datos de procedimientos

En la Figura 6.1, se muestra el esquema relacional que describe con detalle el diseño del modelo de datos de los procedimientos. A este esquema es necesario unirle el esquema de la Figura 5.1, puesto que dichas tablas también serán consultadas por dichos procedimientos.

A continuación se describen las principales entidades lógicas de la aplicación Web:

- **RssContext**: En dicha tabla se almacena el identificador de la última entrada analizada de cada fuente RSS favorita de un determinado bot. Cada registro se compone de los siguientes campos:
  - bot_id: El nombre del bot. Indica a que bot pertenece la fuente analizada (FK).
  - last_rss: La dirección URL de la última entrada analizada (I1).
  - feed: La fuente a la que pertenece la última entrada analizada (I1).
  - date: La fecha en la que se actualizó por última vez el registro.

- **WriteContext**: En dicha tabla se almacenan los Tweets y ReTweets publicados por el bot. De este modo se tiene un control del número de Tweets y de ReTweets publicados por el bot, y se evita que el bot publique o haga un ReTweet de un estado que se ha publicado con anterioridad.
  - bot_id: El nombre del bot. Indica a que bot pertenece el mensaje almacenado (FK).
  - type: Indica el tipo de mensaje que se ha publicado (RSS: publicación Rss, RT: ReTweet, N: Normal, FF. Follow on Friday, TGIF: Thanks God It's Friday, FD: Comida, GN: Good nights y GM: Good Morning).
  - text: Almacena el texto del mensaje (I1).
  - link: Almacena el link del mensaje (si lo tuviera).
  - hashtag: Almacena el hashtag del mensaje (si lo tuviera)
  - user: Si el mensaje es de tipo REPLY, para almacenar el usuario al que se le ha enviado el REPLY.
  - date: La fecha en la que se ha enviado el mensaje.

- **SearchHashtagContext**: Almacena el último id de una búsqueda realizada con el API Search de un hashtag dado. Este valor se almacena para leer en la próxima búsqueda a partir de este punto, y que el API de Twitter no devuelva estados antiguos.
  - hashtag: es el hashtag sobre el que se ha realizado la búsqueda en Twitter (PK).
  - last_id: El id del último estado encontrado.
  - date: La fecha en la que se actualizó el registro.

- **FollowingContext**: Almacena la lista de Following en Twitter de un bot.

  - bot_id: El nombre del bot. Indica a que bot pertenece un determinado Following (FK).
  - name_following: Es el nombre del Following de un determinado bot (I1).
  - date: La fecha en la que se ha añadido como Following un determinado usuario.

- **FollowerContext**: Almacena la lista de Follower en Twitter de un bot.
  - bot_id: El nombre del bot. Indica a que bot pertenece un determinado Follower (FK).
  - name_follower: Es el nombre del Follower de un determinado bot (I1).
  - date: La fecha en la que se ha añadido como Follower un determinado usuario.

- **UnfollowingContext**: Almacena la lista de usuarios que se han dejado de seguir por un bot.
  - bot_id: El nombre del bot. Indica a que bot pertenece un determinado Unfollowing (FK).
  - name_unfollowing: Es el nombre del Following que se ha dejado de seguir en Twitter por un determinado bot (I1).
  - caused_by: El motivo por el que se ha dejado de seguir a un determinado usuario en Twitter (AP: A través de la aplicación, NP: Por no complir el perfil de Following, NF: Por no corresponder al Follow realizado por el bot).
  - count: El número de veces que se ha dejado de seguir a un usuario.
  - date: La fecha en la que se ha actualizado el registro.

- **PolitenessTweets**: Almacena los mensajes de cortesía que pueden ser enviados por el bot.
  - text: Almacena el texto del mensaje de cortesía.
  - type: Almacena el tipo del mensaje (NF: Nuevo Follower, RT: Agradecer RT, FF: Agradecer FF)
  - lang: El lenguaje del mensaje.

- **PolitenessContext**: Almacena los Mensajes Directos de cortesía por motivo de Nuevo Follower (NF) que ha enviado un bot.
  - bot_id: El nombre del bot. Indica a que bot pertenece el mensaje enviado (FK).
  - to_user: El nombre del usuario al que se le ha enviado un mensaje directo de cortesía.
  - text: El texto del mensaje que se ha enviado.

  - type: El tipo de mensaje que se ha enviado (NF: New Follower, FF: Follow on Friday).
  - date: La fecha en la que se ha actualizado el registro.

- **PolitenessRTContext**: Almacena los mensajes de cortesía dirigidos a aquellos usuarios que han realizado RT de los mensajes del bot.
  - bot_id: El nombre del bot. Indica a que bot pertenece el mensaje enviado (FK).
  - to_user: El nombre del usuario al que se le ha dirigido un mensaje de cortesía.
  - text: El texto del mensaje que se ha publicado.
  - status: El mensaje del bot al que se ha realizado RT, para no duplicar agradecimientos.
  - date: La fecha en la que se ha actualizado el registro.

- **TGIFTweets**: Almacena los mensajes de tipo TGIF que podrán ser publicados por un bot.
  - text: Es el texto del mensaje.
  - lang: El lenguaje del mensaje.

- **FamilyContext**: Almacena el contexto familiar de un determinado bot. Esto es, los familiares de los que hablará en Twitter.
  - bot_id: El nombre del bot. Indica a que bot pertenece el contexto familiar (FK).
  - pariente: El pariente sobre el que el bot hablará en Twitter.
  - cardinalidad: La cardinalidad del pariente (S: singular, P. plural)
  - genero: El genero del pariente (sólo si la cardinalidad es singular): (M: masculino, F: femenino)

- **ActivityFamilyContext**: Almacena el contexto de actividades familiares, para que ciertas actividades rutinarias, como por ejemplo: Jugar al pádel. se hagan siempre con el mismo familiar.
  - bot_id: El nombre del bot. Indica a que bot pertenece el contexto de la actividad familiar (FK).
  - actividad: El nombre de la actividad.
  - pariente: El pariente con el que se realiza la actividad rutinaria.
  - cardinalidad: La cardinalidad del pariente (S: singular, P. plural)
  - genero: El genero del pariente (sólo si la cardinalidad es singular): (M: masculino, F: femenino)

- **ActivityWriteContext**: Almacena el contexto de actividades familiares publicadas en Twitter, para que ciertas actividades no se repitan hasta que no se cumplan su periodicidad, por ej: Ir al médico cada 6 meses, y para que el bot no realice publicaciones mientras se encuentra realizando una actividad familiar.
  - bot_id: El nombre del bot. Indica a que bot pertenece el contexto de publicación de la actividad familiar (FK).
  - actividad: El nombre de la actividad.
  - date: La fecha en la que se ha realizado la publicación de la actividad.
  - t_restante: Es el tiempo que queda para que finalice la actividad familiar.

- **RTContext**: Almacena el contexto de los RTs realizados a la lista de Following del bot, para no realizar demasiados RTs al mismo Following y evitar que dicho usuario bloquee al bot.
  - bot_id: El nombre del bot que ha realizado el ReTweet a un Following (FK).
  - autor_RT: El nombre del Following al que se le ha realizado ReTweet.
  - date: La fecha en la que se ha realizado el ReTweet.

- **SearchRTContext**: Almacena el último id de una búsqueda con Search API de los RTs realizados al bot por el método tradicional . Este valor se almacena para obtener Tweets a partir del último estado encontrado, y que el API de Twitter no devuelva estados antiguos.
  - last_id: El id del último estado leido.
  - date: La fecha en la que se actualizó el registro.

- **GoodMorningTweets**: Almacena los mensajes de tipo "Buenos días" que podrán ser publicados por el bot.
  - text: Es el texto del mensaje.
  - type. Es el tipo de mensaje (ANT: parte anterior, POST: parte posterior, FIN: parte final del mensaje).
  - dow: El día de la semana (1-7 o *: para cualquier día de la semana).
  - lang: Es el lenguaje del mensaje.

- **GoodNightsTweets**: Almacena los mensajes de tipo "Buenas noches" que podrán ser publicados por el bot.
  - text: Es el texto del mensaje.
  - type. Es el tipo de mensaje (ANT: parte anterior, POST: parte posterior, FIN: parte final del mensaje).

- lang: Es el lenguaje del mensaje.

- **FoodTweets**: Almacena los mensajes de comida que podrán ser publicados por el bot.
  - text: Es el texto del mensaje.
  - type. Es el tipo de mensaje (ANTC: parte anterior comida, ANTP: parte anterior postre, CMD: comida, PST: postre, POST: parte posterior, FIN: parte final del mensaje).
  - lang: Es el lenguaje del mensaje.

- **FFTweets**: Almacena los mensajes de tipo #FF que podrán ser publicados por el bot.
  - text: Es el texto del mensaje.
  - type. Es el tipo de mensaje (ANT: parte anterior, POST: parte posterior, FIN: parte final del mensaje).
  - cardinalidad: La cardinalidad del mensaje (S: singular, P: plural), para saber si se menciona a varios usuarios en el mensaje o solo a uno.
  - lang: Es el lenguaje del mensaje.

- **FFContext**: Almacena los mensajes de tipo #FF enviados por un determinado usuario, para no realizar #FF repetidos.
  - bot_id: El nombre del bot que ha enviado un mensaje de tipo #FF (FK).
  - name_following: Es el nombre del usuario al que se ha mencionado en un mensaje de tipo #FF.
  - date: La fecha en la que se ha enviado el mensaje de tipo #FF.

- **PolitenessFFContext**: Almacena los mensajes de agradecimiento de mensejes de tipo #FF en los que se menciona al bot.
  - bot_id: El nombre del bot que ha enviado un mensaje de agradecimiento a un #FF (FK).
  - to_user: El nombre del usuario al que se le ha dirigido un mensaje de cortesía.
  - text: El texto del mensaje que se ha publicado.
  - status: El mensaje en el que se realizaba #FF al bot, para no duplicar agradecimientos.
  - date: La fecha en la que se ha añadido el registro.

6.4. Diseño de la lógica de negocio de los procedimientos

Un procedimiento es una clase Python encargada de hacer una determinada tarea que permitirá la correcta comunicación en Twitter del bot. Todos los procedimientos serán capaces de conectarse al servidor donde se encuentra la Base de Datos mediante el conector MySQLdb 3.4.3 y las clases necesarias tendrán acceso al API de Twitter 3.4.7, al API de bit.ly 3.4.6 y al parseador de fuentes 3.4.5. Los procedimientos son independientes entre sí

A continuación se muestran en forma de tabla, junto con el diagrama de flujo correspondiente, cada uno de los procedimientos diseñados en función de los requisitos previamente definidos por el usuario 4.3:

Cuadro 6.3: Procedimiento (I): reset_RSS

Procedimiento	***reset_RSS***
Objetivo	Eliminar contenido obsoleto de fuentes RSS del fichero de contenidos de cada bot. Relacionado con el requisito 4.3.5
Estrategias	Ninguna
Contexto	Ninguno.
Descripción procedimiento	Por cada bot: ▪ Se abre el fichero que almacena los contenidos interesantes recogidos de cada fuente RSS. ▪ Se recorre el fichero y se eliminan aquellos contenidos que hayan sido publicados hace más de un día.

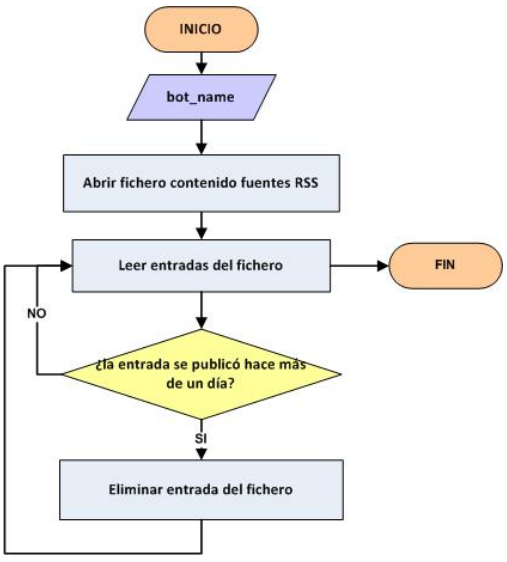


Figura 6.2: Diagrama de flujo: reset_RSS

Cuadro 6.4: Procedimiento (II): reset_hashtags

Procedimiento	***reset_hashtags***
Objetivo	Eliminar Hashtags obsoletos de la lista de Hashtags favoritos del bot. Relacionado con el requisito 4.3.4
Estrategias	Ninguna
Contexto	Ninguno.
Descripción procedimiento	Por cada bot: ▪ Se obtiene la lista de Hashtags favoritos del bot de la Base de datos. ▪ Por cada Hashtag se comprueba la fecha de su última actualización. ▪ Si el Hashtag lleva sin actualizarse más de 15 días se elimina de la lista de Hashtags favoritos del bot.

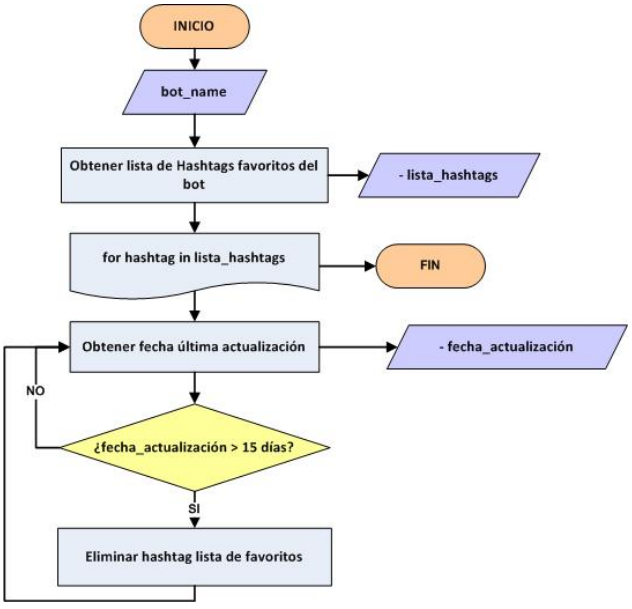


Figura 6.3: Diagrama de flujo: reset_hashtags

Cuadro 6.5: Procedimiento (III): search_RT

Procedimiento	***search_RT***
Objetivo	Agradecer RTs realizados por otros usuarios a los estados del bot y seguir a dichos usuarios si cumplen el perfil de Following establecido. Relacionado con el requisito 4.3.2
Estrategias	▪ Estr. de Cortesía: Que establecerá si el bot enviará o no, mensajes de agradecimiento. ▪ Estr. de Following: Que establecerá si el bot seguirá o no a los usuarios que cumplan el perfil de Following. ▪ Estr. de Publicación: Que establecerá el horario de publicación del bot a través del Timeline.
Contexto	Se almacenarán los mensajes de agrademiento enviados.
Descripción procedimiento	Por cada bot: ▪ Se obtiene la lista de usuarios que han realizado RTs de las publicaciones del bot. ▪ Se comprueba si el bot implementa la estrategia de Cortesía de enviar mensajes de agradecimiento. ▪ Si el bot implementa la estrategia y se decide aleatoriamente enviar un mensaje de agradecimiento, se obtiene el modo en que se enviará dicho mensaje: • Si se envía a través del Timeline, se ha de comprobar si el bot se encuentra dentro del horario de publicación. • Si se envía a través de DM, se enviará un mensaje directo a cada uno de los usuarios que hayan realizado RT. ▪ Se almacena el contexto de agradecimiento para no enviar agradecimientos repetidos. ▪ Si el bot implementa la estrategia de Following, se comenzará a seguir a aquellos usuarios que cumplan el perfil de Following.

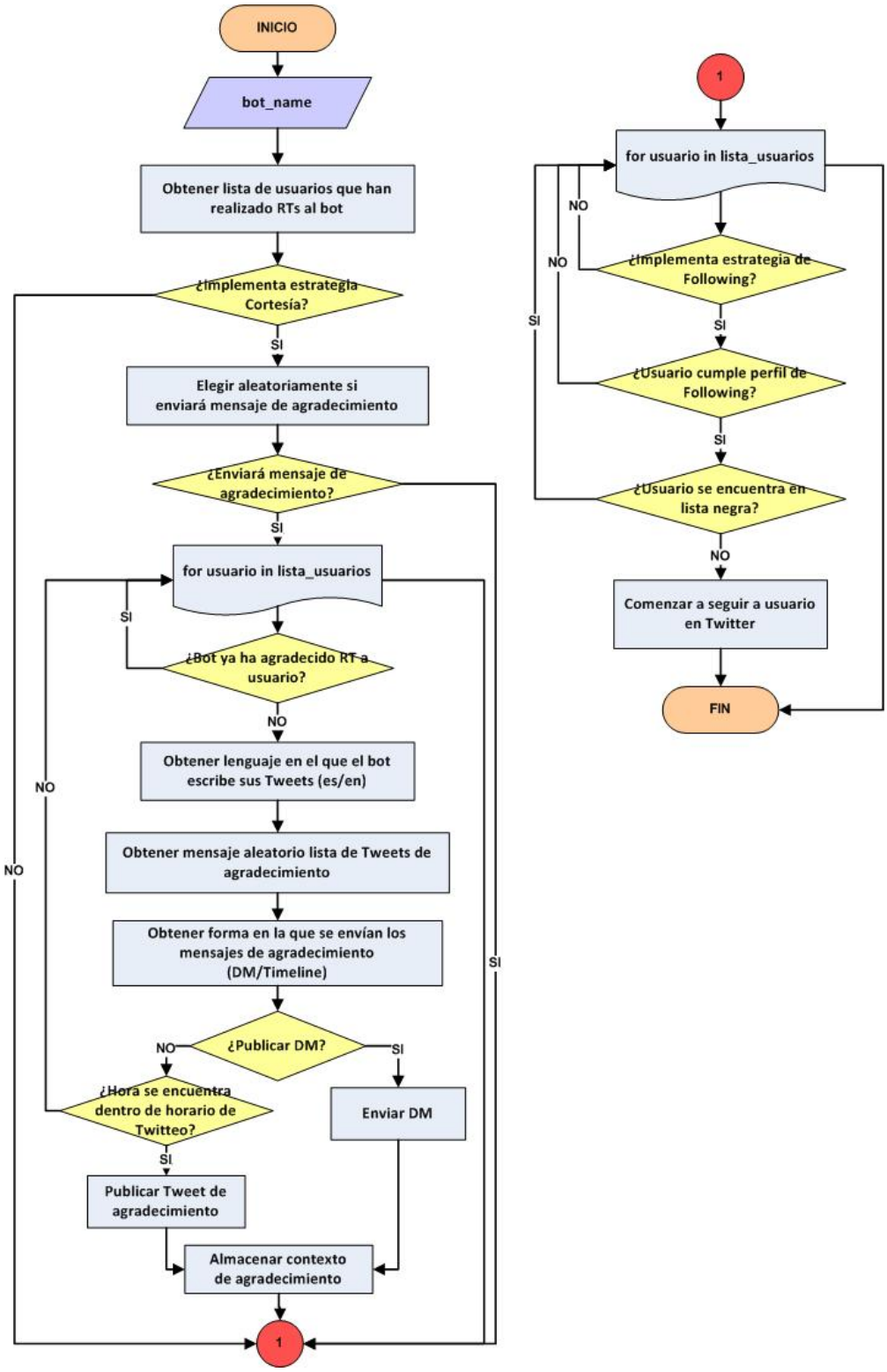


Figura 6.4: Diagrama de flujo: search_RT

Cuadro 6.6: Procedimiento (IV): read_TL

Procedimiento	***read_TL***
Objetivo	Leer el Timeline de la lista de Following del bot para realizar RT a aquellos estados que se consideren interesantes. Relacionado con el requisito 4.3.3
Estrategias	▪ Estr. de Publicación: Que establecerá el horario de publicación del bot a través del Timeline, así como el número máximo de ReTweets que podrá publicar el bot a lo largo del día.
Contexto	Se almacenará el id del último estado leído, y RTs realizados por el bot.
Descripción procedimiento	Por cada bot: ▪ Si se decide aleatoriamente leer el Timeline de los usuarios que sigue, se obtienen los últimos estados del Timeline (mensajes normales y RTs). ▪ Por cada estado encontrado se comprueba si tiene URL. ▪ Si tiene URL, se comprueba su relevancia: ▪ Si es una URL relevante se añade a una lista de mensajes interesantes. ▪ Una vez procesados todos los mensajes leidos a través del Timeline, se escoge un estado de entre la lista de estados interesantes encontrados. ▪ Si el bot se encuentra en horario de publicación y todavía no se han publicado el número máximo de RTs permitidos, se comprueba si el mensaje se encuentra en el contexto de RTs realizados por el bot. ▪ Si el mensaje no se encuentra en el contexto, se realiza RT del estado seleccionado y se almacena en el contexto, para no duplicar RTs.

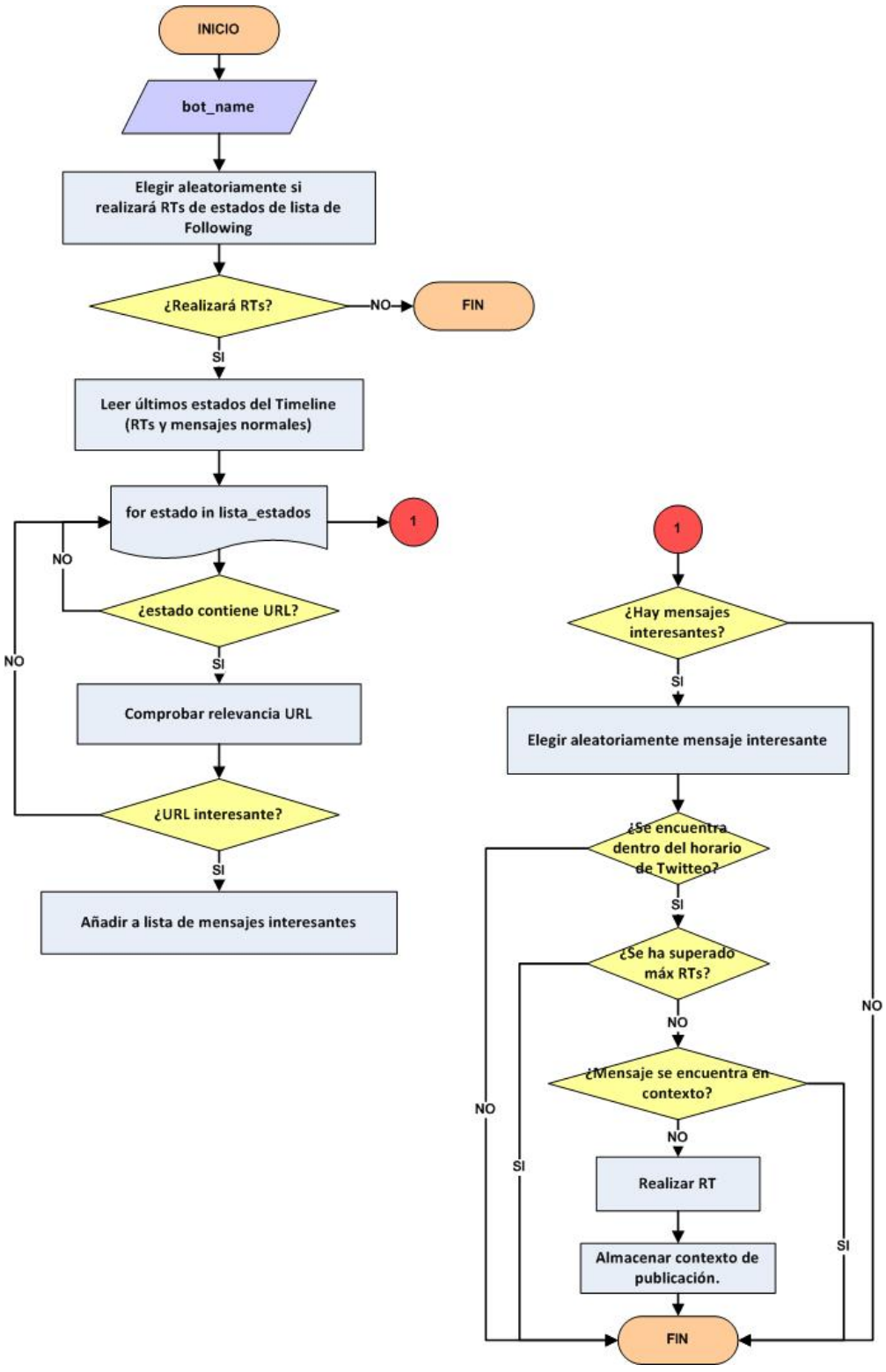


Figura 6.5: Diagrama de flujo: read_TL

Cuadro 6.7: Procedimiento (V): search_hashtags

Procedimiento	***search_hashtags***
Objetivo	Buscar publicaciones en Twitter que contengan los Hashtags favoritos del bot para realizar RT a aquellos estados que se consideren interesantes y seguir a los usuarios que hagan uso de dichos Hashtags si cumplen el perfil de Following establecido. Relacionado con el requisito 4.3.4
Estrategias	- Estr. de Publicación: Que establecerá el horario de publicación del bot a través del Timeline, así como el número máximo de ReTweets que podrá publicar el bot a lo largo del día. - Estr. de Following: Que establecerá si el bot seguirá o no a los usuarios que cumplan el perfil de Following.
Contexto	Se almacenará el id del último estado leído, y RTs realizados por el bot.
Descripción procedimiento	Por cada bot: ▪ Se obtienen los hashtags favoritos del bot. Por cada hashtag: • Se hace una búsqueda en Twitter para obtener lista de mensajes y lista de usuarios que hacen uso del hashtag, junto con la lista de hashtags que mas se repiten. • De entre la lista de hashtags que más se repiten coger el que más se repite. Si cumple condiciones se añade a la lista de hashtags favoritos. • De entre la lista de mensajes encontrados se escoge el más repetido y se escoge aleatoriamente si realizar RT o no. • Si el bot se encuentra en horario de publicación y todavía no se han publicado el número máximo de RTs permitidos, se comprueba si el mensaje se encuentra en el contexto de RTs realizados por el bot, se realiza RT del estado seleccionado y se almacena en el contexto, para no duplicar RTs. • Si el bot implementa la estrategia de Following, se comenzará a seguir a aquellos usuarios encontrados que cumplan el perfil de Following y que no se encuentren en lista negra.

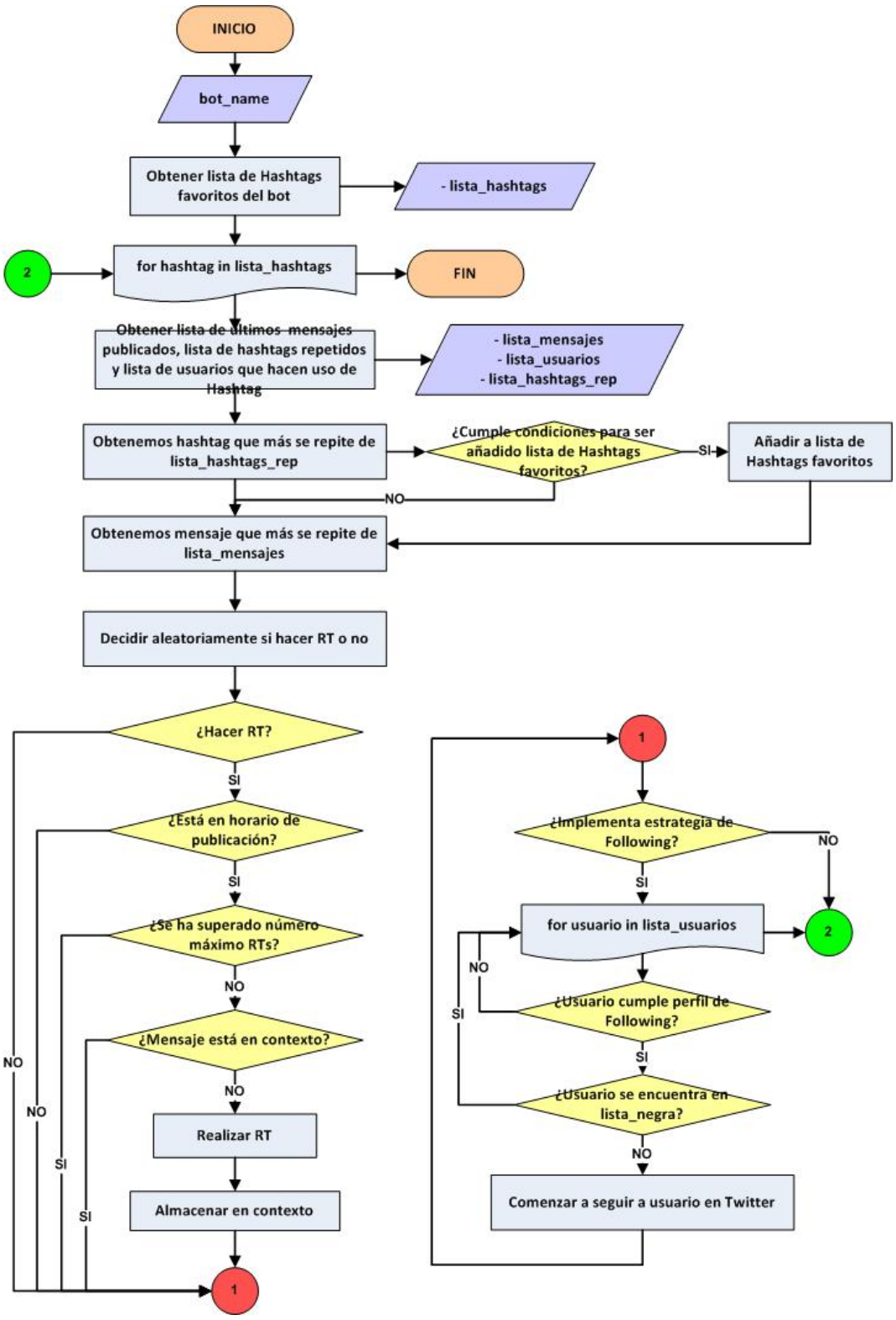


Figura 6.6: Diagrama de flujo: search_hashtags

Cuadro 6.8: Procedimiento (VI): read_rss

Procedimiento	***read_rss***
Objetivo	Buscar publicaciones interesantes en las fuentes RSS favoritas del bot para que posteriormente el bot pueda realizar publicaciones en Twitter. Relacionado con el requisito 4.3.5
Estrategias	Ninguna
Contexto	Última URL consultada de cada fuente.
Descripción procedimiento	Por cada bot: ▪ Se obtienen la lista de fuentes favoritas del bot. Por cada fuente: • Se obtiene el hashtag asociado (si lo tuviera). • Se obtienen las últimas publicaciones de la fuente (A partir de la última URL almacenada). • Por cada publicación: ◦ Se obtiene su Título, URL, fecha de publicación (feedparser) y número de clicks (API bit.ly). ◦ Se decide si es una publicación interesante mediante un algoritmo que tiene en cuenta los minutos que han pasado desde su publicación y el número de clicks. ◦ Si es una publicación interesante se almacena su información en un fichero de texto (título, URL, hashtag asociado) • Se almacena la última URL consultada.

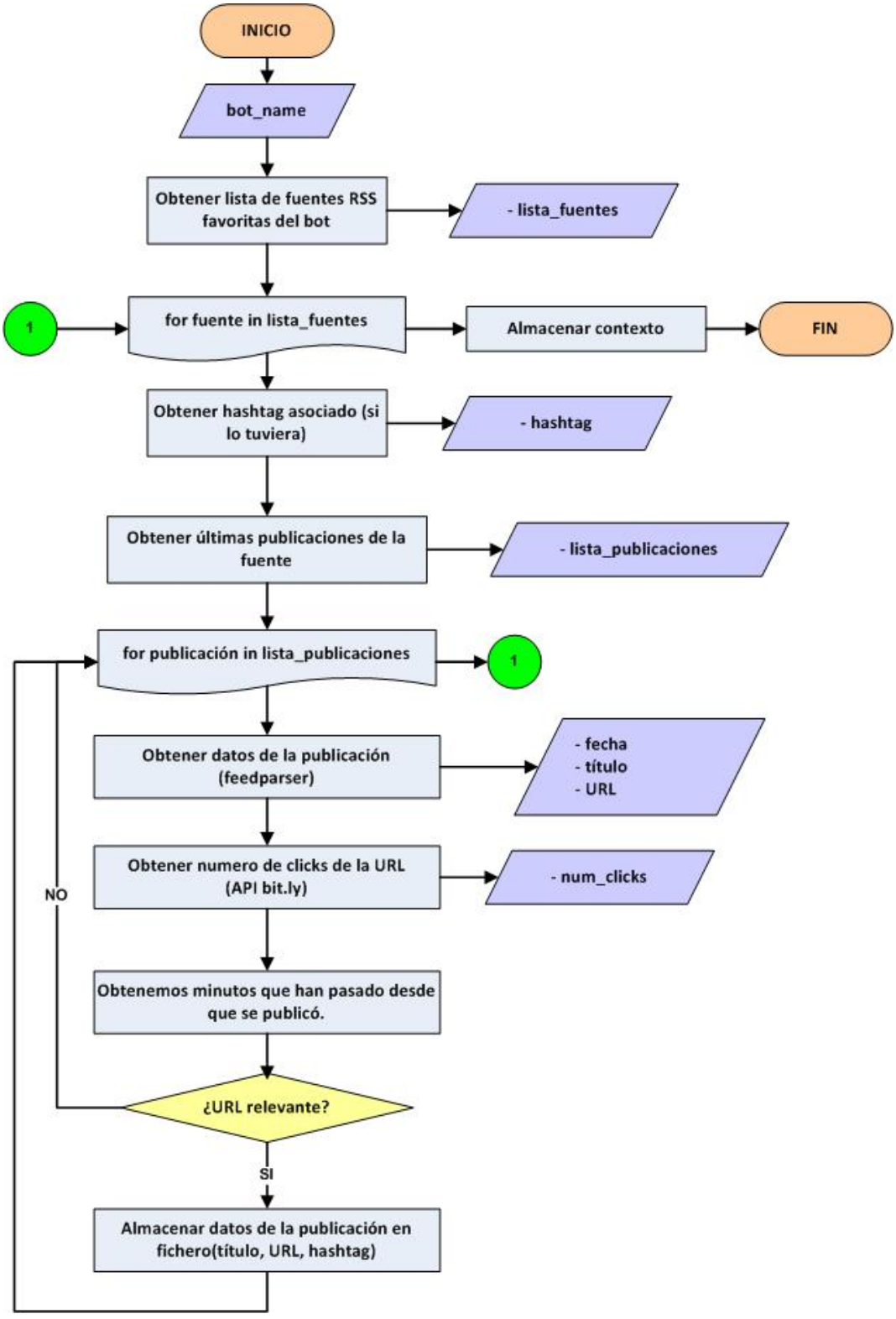


Figura 6.7: Diagrama de flujo: read_rss

Cuadro 6.9: Procedimiento (VII): write_goodmorning

Procedimiento	***write_goodmorning***
Objetivo	Enviar mensajes de tipo "Buenos días" a través del Timeline del bot. Relacionado con el requisito 4.3.6
Estrategias	- Estr. de Publicación: Que establecerá el horario de publicación del bot a través del Timeline, así como el número máximo de Tweets que podrá publicar el bot a lo largo del día. - Estr. de Personalidad: Que establecerá si el bot enviará mensajes de "Buenos días" a sus seguidores a través de su Timeline.
Contexto	Último Tweet publicado.
Descripción procedimiento	Por cada bot: ▪ Se comprueba si el bot implementa la estrategia de Personalidad de enviar mensajes de tipo "Buenos días" a sus seguidores a través de su Timeline. Si se implementa la estrategia: • Se comprueba que no se haya enviado todavía un mensaje de tipo "Buenos días". • Decidir aleatoriamente si se enviará un mensaje de "Buenos días". • Se obtiene el lenguaje en el que publica el bot y la lista de mensajes de tipo "Buenos días" de la Base de datos. • Se escoge un mensaje aleatorio de la lista de mensajes. • Se obtiene el modo en el que se enviarán los mensajes de "Buenos días" (Personalizados/Generales). • Si se envían de modo Personalizados, se obtiene la lista de últimos Followers incorporados y se añade al mensaje. • Si el bot se encuentra en horario de publicación y todavía no se han publicado el número máximo de Tweets permitidos, se publica el mensaje seleccionado y se almacena en el contexto, para no duplicar Tweets.

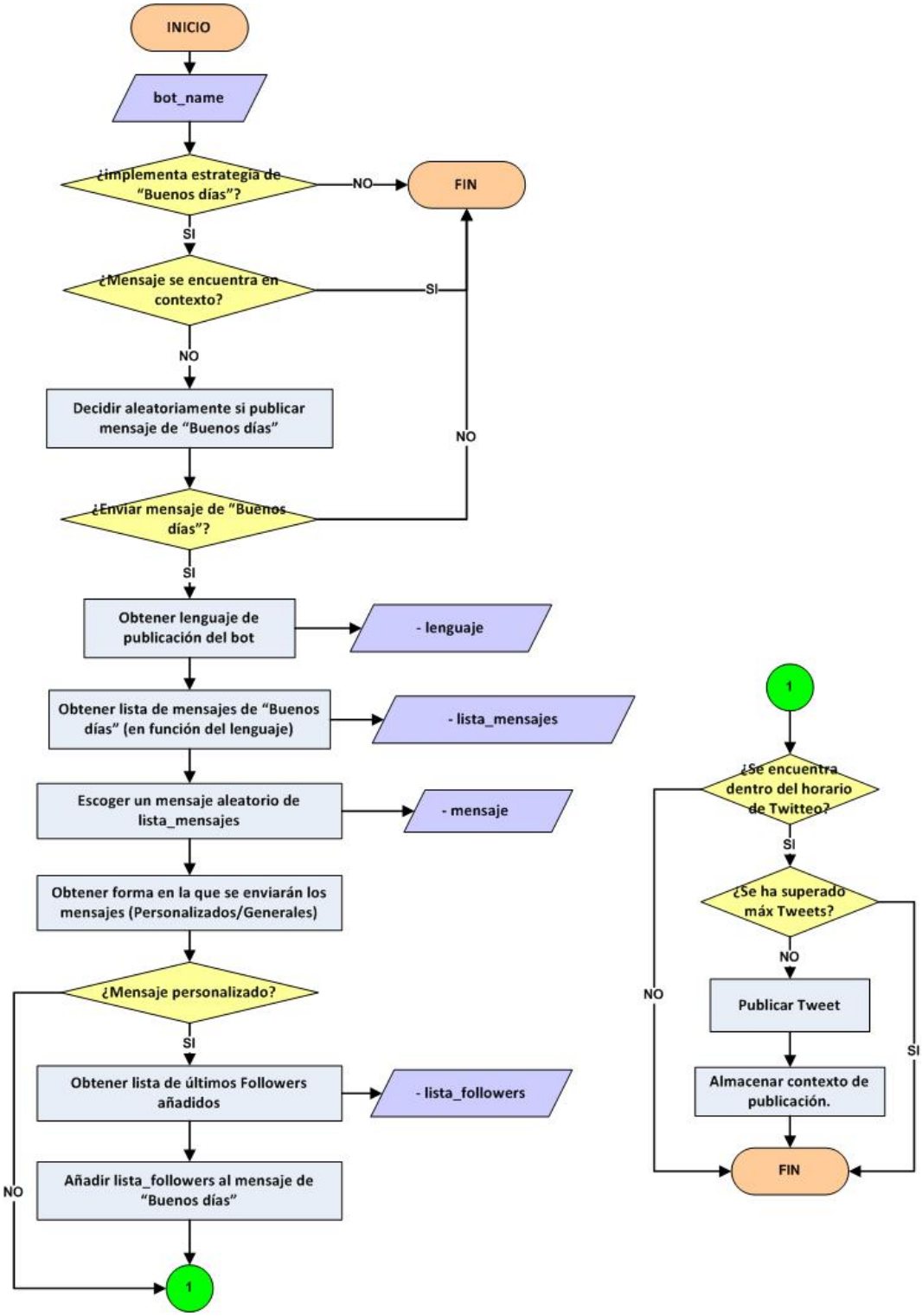


Figura 6.8: Diagrama de flujo: write_goodmorning

Cuadro 6.10: Procedimiento (VIII): write_goodnights

Procedimiento	***write_goodnights***
Objetivo	Enviar mensajes de tipo "Buenas noches" a través del Timeline del bot. Relacionado con el requisito 4.3.6
Estrategias	- Estr. de Publicación: Que establecerá el horario de publicación del bot a través del Timeline, así como el número máximo de Tweets que podrá publicar el bot a lo largo del día. - Estr. de Personalidad: Que establecerá si el bot enviará mensajes de "Buenas noches" a sus seguidores a través de su Timeline.
Contexto	Último Tweet publicado.
Descripción procedimiento	Por cada bot: ▪ Se comprueba si el bot implementa la estrategia de Personalidad de enviar mensajes de tipo "Buenas noches" a sus seguidores a través de su Timeline. Si se implementa la estrategia: • Se comprueba que no se haya enviado todavía un mensaje de tipo "Buenas noches". • Decidir aleatoriamente si se enviará un mensaje de "Buenas noches". • Se comprueba si quedan menos de 60 minutos para que finalice el horario de publicación del bot. • Se obtiene el lenguaje en el que publica el bot y la lista de mensajes de tipo "Buenas noches" de la Base de datos. • Se escoge un mensaje aleatorio de la lista de mensajes. • Si el bot se encuentra en horario de publicación y todavía no se han publicado el número máximo de Tweets permitidos, se publica el mensaje seleccionado y se almacena en el contexto, para no duplicar Tweets.

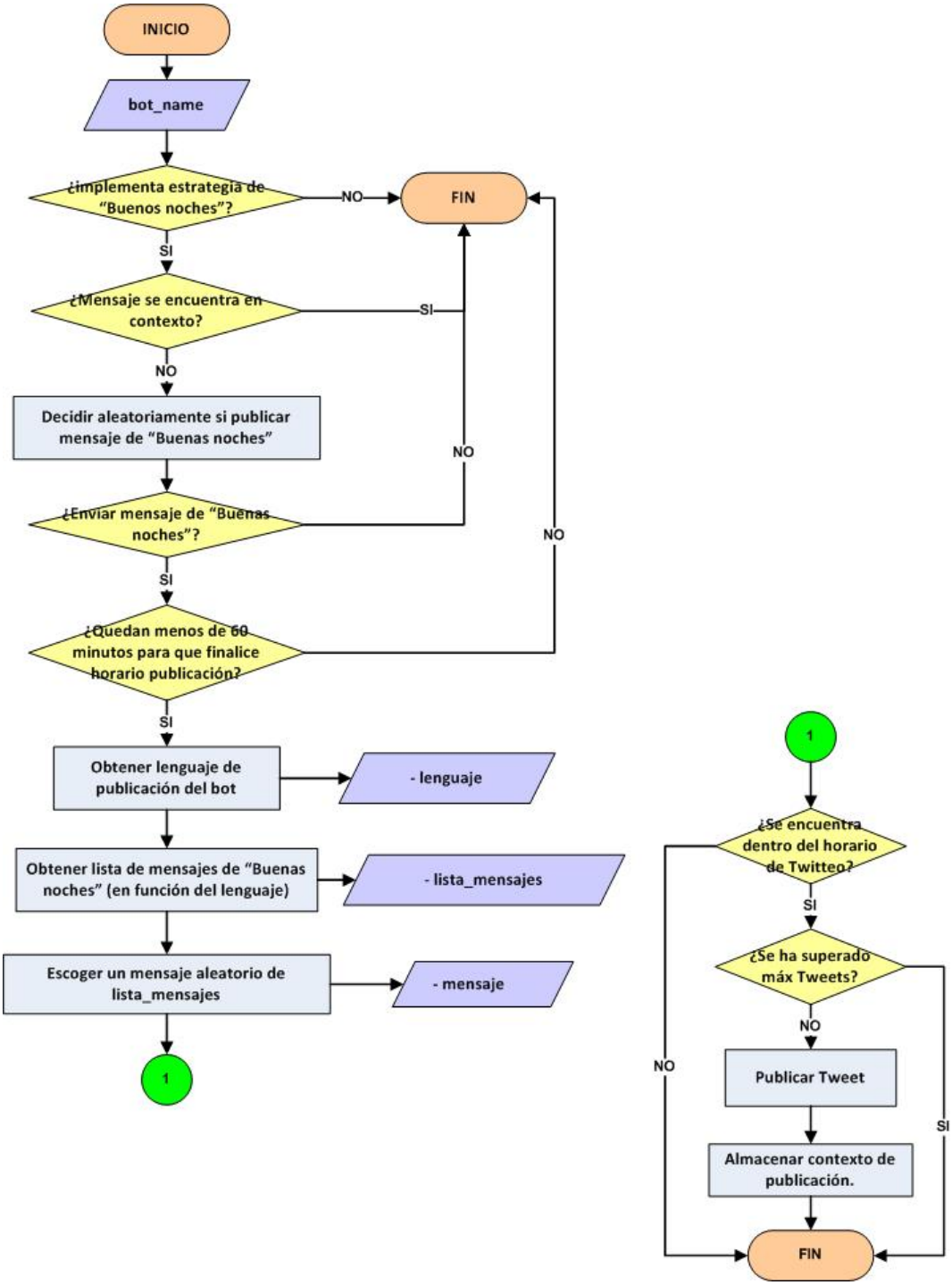


Figura 6.9: Diagrama de flujo: write_goodnights

Cuadro 6.11: Procedimiento (IX): write_food

Procedimiento	***write_food***
Objetivo	Enviar mensajes para informar a los seguidores del bot sobre qué está comiendo a través de su Timeline. Relacionado con el requisito 4.3.6
Estrategias	- Estr. de Publicación: Que establecerá el horario de publicación del bot a través del Timeline, el horario de publicación de comidas, así como el número máximo de Tweets que podrá publicar el bot a lo largo del día. - Estr. de Personalidad: Que establecerá si el bot enviará mensajes para informar a sus seguidores sobre qué está comiendo a través de su Timeline.
Contexto	Último Tweet publicado.
Descripción procedimiento	Por cada bot: ▪ Se comprueba si el bot implementa la estrategia de Personalidad de enviar mensajes a sus seguidores para informar sobre qué está comindo a través de su Timeline. Si se implementa la estrategia: • Se comprueba que no se haya enviado todavía un mensaje de comida. • Decidir aleatoriamente si se enviará un mensaje de comida. • Se comprueba si se encuentra dentro del horario establecido de comidas. • Se obtiene el lenguaje en el que publica el bot y la lista de mensajes de comidas de la Base de datos. • Se escoge un mensaje aleatorio de la lista de mensajes. • Si el bot se encuentra en horario de publicación y todavía no se han publicado el número máximo de Tweets permitidos, se publica el mensaje seleccionado y se almacena en el contexto, para no duplicar Tweets.

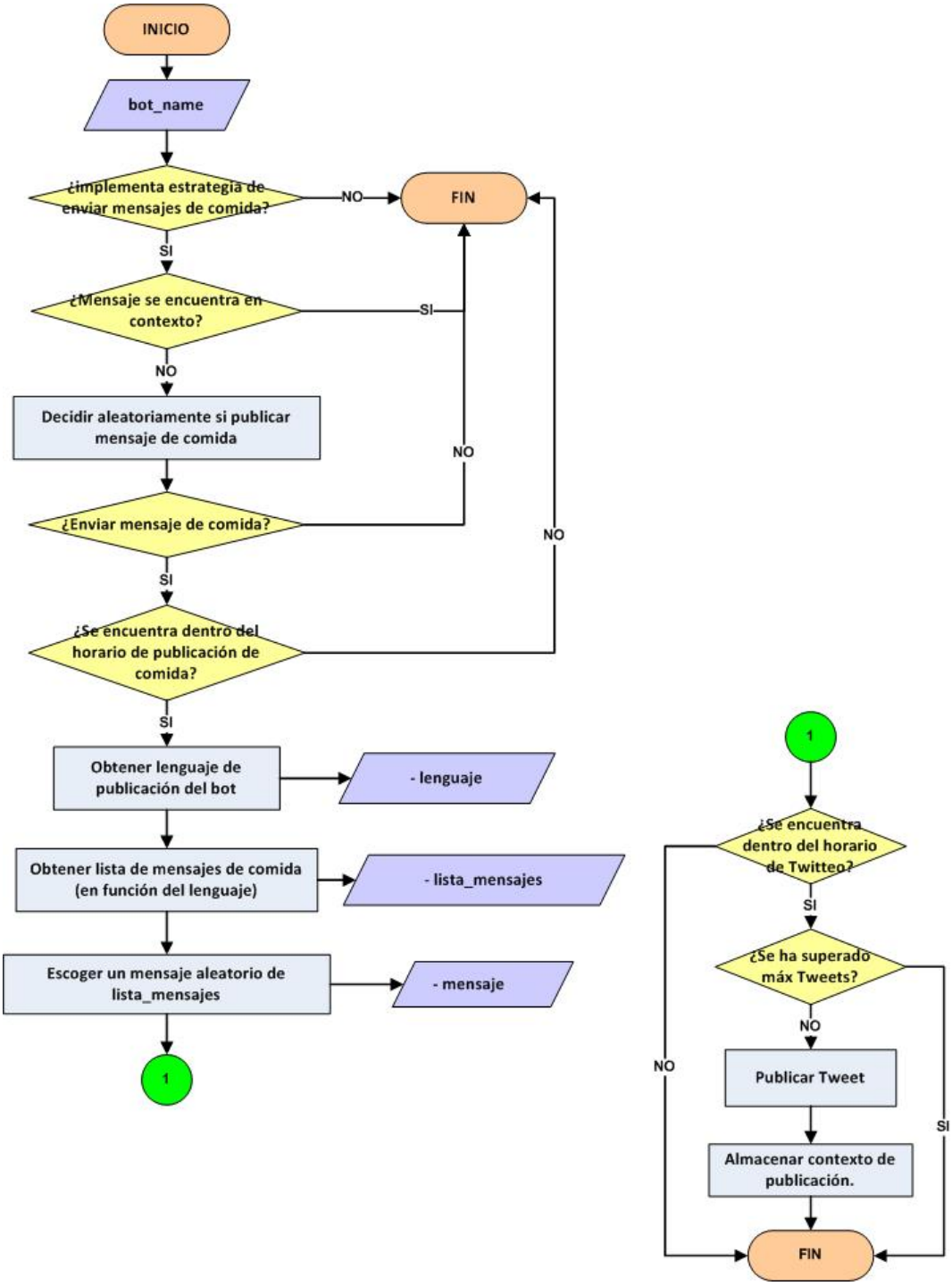


Figura 6.10: Diagrama de flujo: write_food

Cuadro 6.12: Procedimiento (X): write_family

Procedimiento	***write_family***
Objetivo	Enviar mensajes para informar a los seguidores del bot sobre las actividades que está realizando con su familia a través de su Timeline. Relacionado con el requisito 4.3.6
Estrategias	- Estr. de Publicación: Que establecerá el horario de publicación del bot a través del Timeline, así como el número máximo de Tweets que podrá publicar el bot a lo largo del día. - Estr. de Personalidad: Que establecerá si el bot enviará mensajes para informar a sus seguidores sobre las actividades que está realizando con su familia a través de su Timeline.
Contexto	Último Tweet publicado.
Descripción procedimiento	Por cada bot: ▪ Se comprueba si el bot implementa la estrategia de Personalidad de enviar mensajes a sus seguidores para informar sobre las actividades que está realizando con su familia a través de su Timeline. Si se implementa la estrategia: • Decidir aleatoriamente si se enviará un mensaje de actividades familiares. • Se obtiene el contexto familiar, esto es, los familiares sobre los que se va a hablar. Si no existe el contexto se establece uno nuevo aleatoriamente. • Se escoge aleatoriamente un familiar del contexto y un tiempo verbal. • Se obtiene una actividad aleatoria de acuerdo al horario actual. • Se comprueba que se haya cumplido la frecuencia para la actividad. • Si el bot se encuentra en horario de publicación y todavía no se han publicado el número máximo de Tweets permitidos, se publica el mensaje seleccionado y se almacena en el contexto, para no duplicar Tweets. • Se almacena el contexto de la actividad.

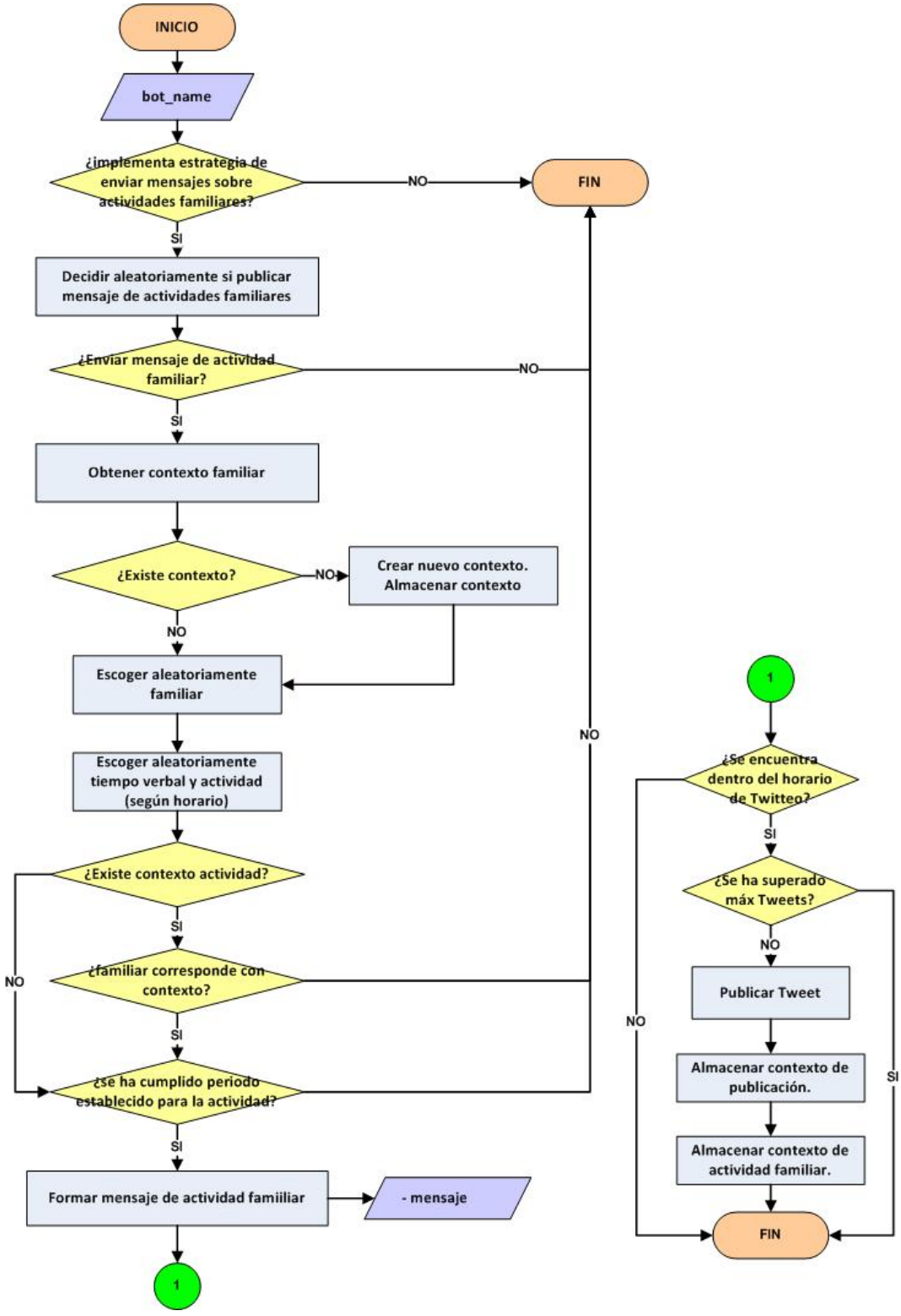


Figura 6.11: Diagrama de flujo: write_family

Cuadro 6.13: Procedimiento (XI): write_TGIF

Procedimiento	***write_TGIF***
Objetivo	Enviar mensajes de tipo TGIF (Thanks God It's Friday) los viernes a través de su Timeline. Relacionado con el requisito 4.3.6
Estrategias	▪ Estr. de Publicación: Que establecerá el horario de publicación del bot a través del Timeline, así como el número máximo de Tweets que podrá publicar el bot a lo largo del día. ▪ Estr. de Personalidad: Que establecerá si el bot enviará mensajes de tipo TGIF los viernes a través de su Timeline.
Contexto	Último Tweet publicado.
Descripción procedimiento	Por cada bot: ▪ Se comprueba si el bot implementa la estrategia de Personalidad de enviar mensajes de tipo TGIF los viernes a través de su Timeline. Si se implementa la estrategia: • Comprobar que no se haya enviado con anterioridad, en el día actual, un mensaje de tipo TGIF. • Decidir aleatoriamente si se enviará un mensaje de tipo TGIF. • Se obtiene el lenguaje en el que escribe el bot. • Se obtiene la lista de mensajes de tipo TGIF de la Base de Datos en función del idioma del bot. • Se escoge aleatoriamente un mensaje de la lista de mensajes. • Si el bot se encuentra en horario de publicación y todavía no se han publicado el número máximo de Tweets permitidos, se publica el mensaje seleccionado y se almacena en el contexto, para no duplicar Tweets.

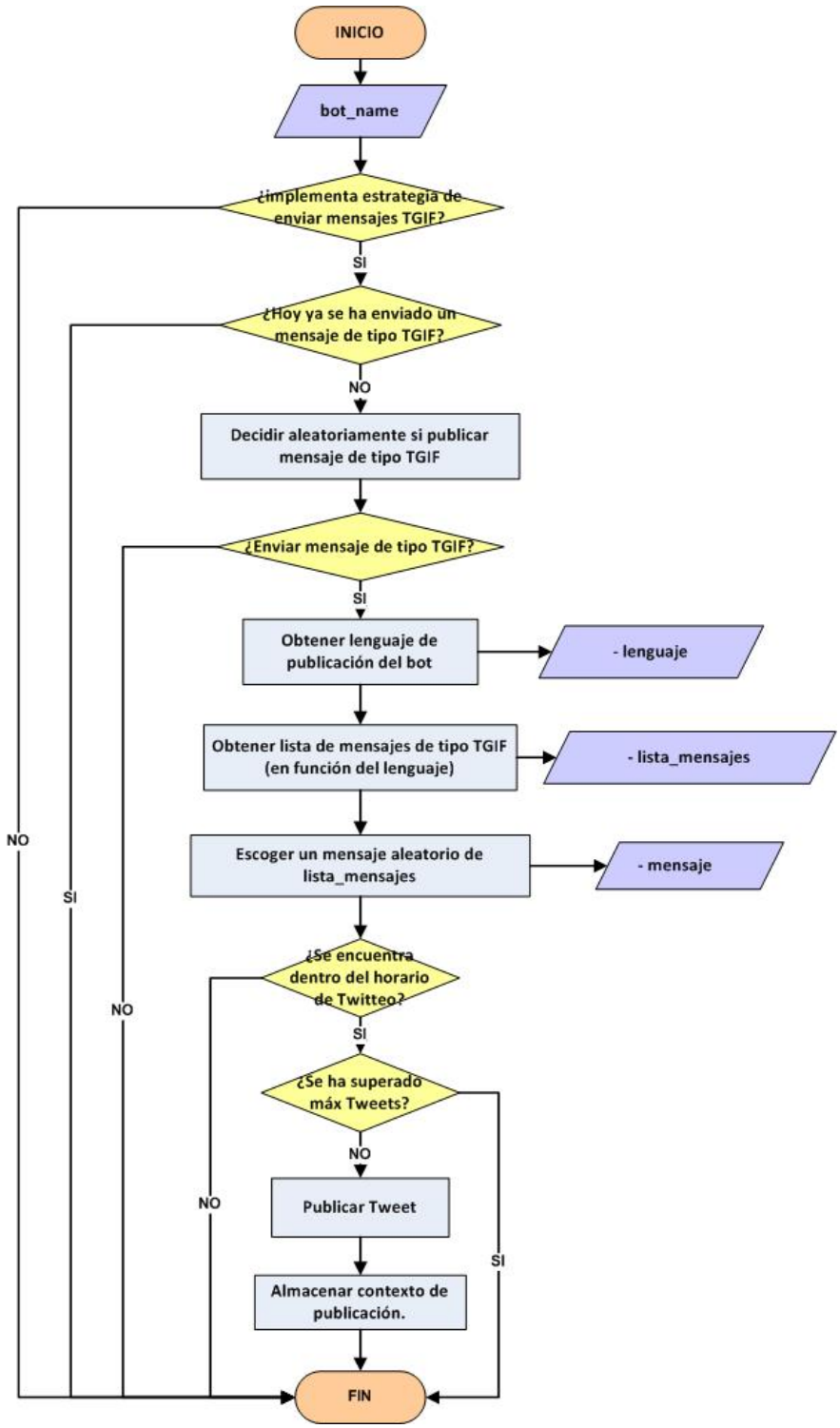


Figura 6.12: Diagrama de flujo: write_TGIF

Cuadro 6.14: Procedimiento (XII): write_RSS

Procedimiento	***write_RSS***
Objetivo	Realizar publicaciones a través de su Timeline de contenidos interesantes encontrados en las fuentes favoritas del bot. Relacionado con el requisito 4.3.5
Estrategias	▪ Estr. de Publicación: Que establecerá el horario de publicación del bot a través del Timeline, así como el número máximo de Tweets que podrá publicar el bot a lo largo del día.
Contexto	Último Tweet publicado.
Descripción procedimiento	Por cada bot: ▪ Decidir aleatoriamente si se publicará contenido de las fuentes RSS favoritas del bot. ▪ Se abre el fichero donde se encuentran almacenados los contenidos interesantes de las fuentes favoritas del bot. ▪ Se escoge aleatoriamente una publicación (Título, URL y Hashtag (si lo tiene)) ▪ Comprobar que dicho contenido no se haya publicado con anterioridad. ▪ Si el bot se encuentra en horario de publicación y todavía no se han publicado el número máximo de Tweets permitidos, se publica el mensaje seleccionado y se almacena en el contexto, para no duplicar Tweets.

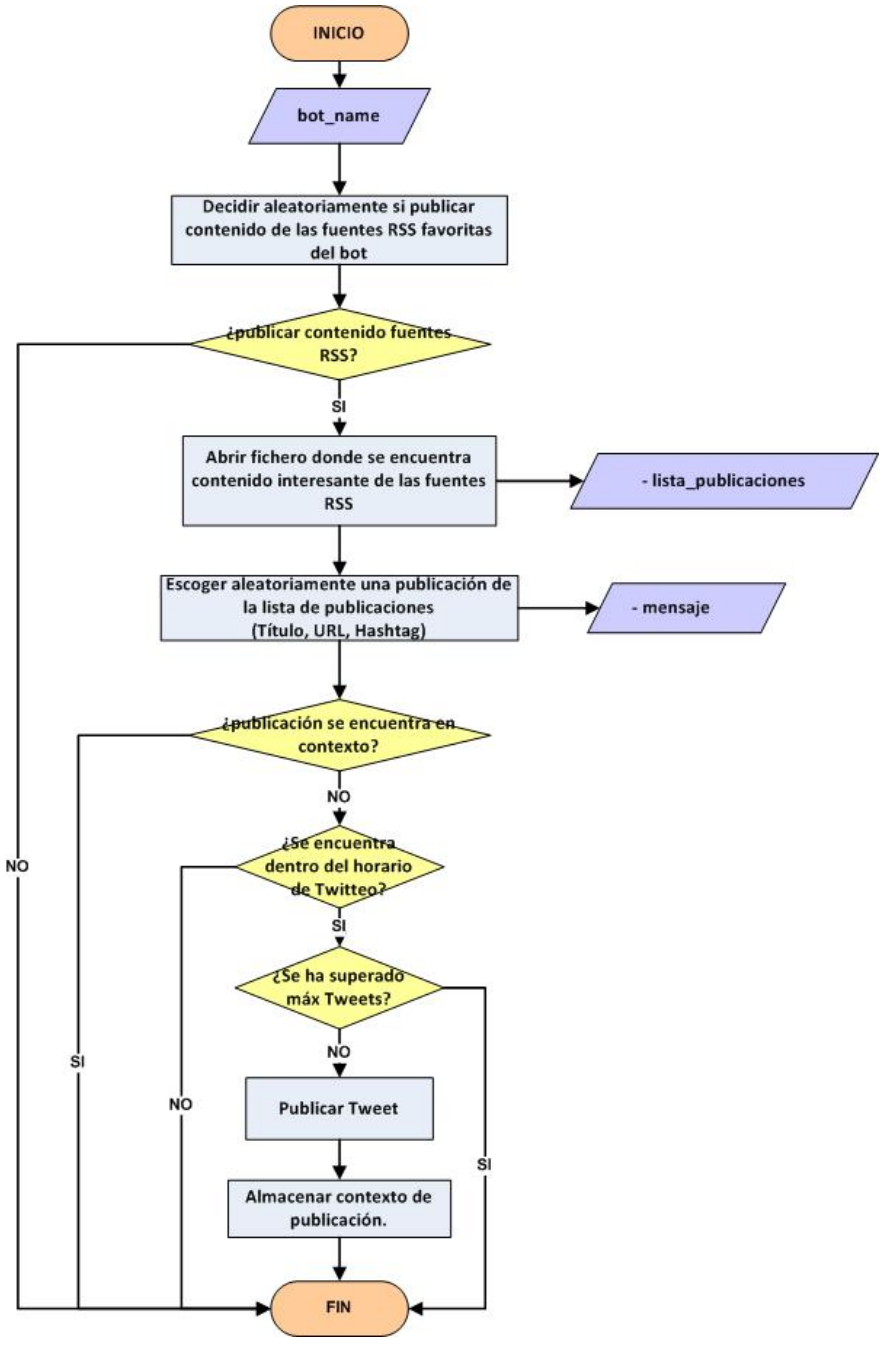


Figura 6.13: Diagrama de flujo: write_RSS

Cuadro 6.15: Procedimiento (XIII): write_FF

Procedimiento	***write_FF***
Objetivo	Enviar mensajes de tipo #FF los viernes a través de su Timeline. Relacionado con el requisito 4.3.6
Estrategias	- Estr. de Publicación: Que establecerá el horario de publicación del bot a través del Timeline, así como el número máximo de Tweets que podrá publicar el bot a lo largo del día. - Estr. de FF: Que establecerá el número máximo de mensajes de tipo #FF que el bot enviará los viernes a través de su Timeline, el número máximo de Menciones que se realizarán en un mismo mensaje de tipo #FF y el perfil que han de cumplir los usuarios a los que se realice #FF.
Contexto	Último Tweet publicado y #FF realizados.
Descripción procedimiento	Por cada bot: ▪ Comprobar si implementa la estrategia de Follow on Friday (FF) y comprobar que no se hayan enviado el máximo de mensajes de tipo #FF permitidos. ▪ Obtener el número máximo de Menciones que se enviarán en cada mensaje. Si es mayor que uno, seleccionar un valor aleatorio entre 1 y el máximo permitido. ▪ Obtener la lista de Following del bot. Seleccionar aleatoriamente usuarios que cumplan el perfil de Following y que no se encuentren en el contexto de FF para ser mencionados en el mensaje. ▪ Si existen usuarios a los que mencionar, se obtiene la lista de mensajes de tipo #FF de la Base de datos, y se selecciona aleatoriamente un mensaje de la lista. Se añade la lista de menciones. ▪ Si el bot se encuentra en horario de publicación y todavía no se han publicado el número máximo de Tweets permitidos, se publica el mensaje seleccionado y se almacena en el contexto, para no duplicar Tweets. ▪ Se almacena la lista de usuarios mencionados en un contexto para no repetir #FF a dichos usuarios.

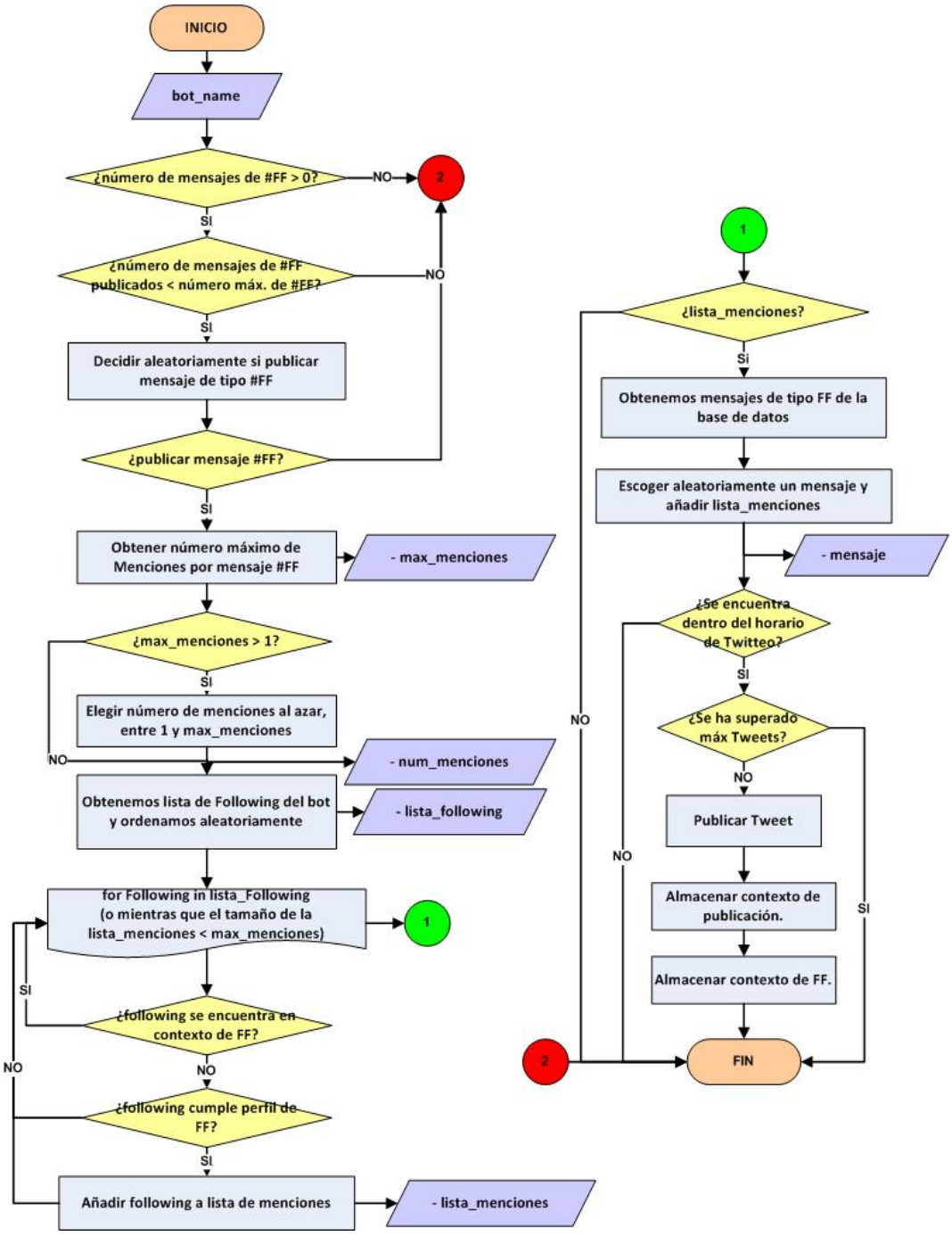


Figura 6.14: Diagrama de flujo: write_FF

Cuadro 6.16: Procedimiento (XIV): search_FF

Procedimiento	***search_FF***
Objetivo	Agradecer mensajes de tipo #FF en los que el bot es mencionado. Relacionado con el requisito 4.3.7
Estrategias	- Estr. de Publicación: Que establecerá el horario de publicación del bot a través del Timeline, así como el número máximo de Tweets que podrá publicar el bot a lo largo del día. - Estr. de Cortesía: Que establecerá si el bot enviará mensajes de agradecimiento a través de su Timeline a los #FF realizados al bot por otros usuarios y si el bot corresponderá a dichos #FF.
Contexto	Último Tweet publicado, #FF realizados y agradecimientos enviados.
Descripción procedimiento	Por cada bot: ▪ Comprobar si implementa la estrategia de agradecer los #FF realizados por otros usuarios. ▪ Buscar mensajes de tipo #FF en los que se mencione al bot y clasificar dichos mensajes en: #FF de correspondencia, agradecimiento a #FF sin correspondencia, agradecimiento a #FF con correspondencia y #FF. Si el mensaje es de tipo #FF se comprueba que no se haya enviado ya un mensaje de agradecimiento al usuario. ▪ Se obtiene la lista de mensajes de agradecimiento a #FF de la Base de datos, y se selecciona aleatoriamente un mensaje de la lista. Si el usuario implementa la estrategia de Cortesía de corresponder los #FF, también se obtiene la lista de mensajes de correspondencia, se selecciona un mensaje de la lista y se añade al mensaje. ▪ Si el bot se encuentra en horario de publicación y todavía no se han publicado el número máximo de Tweets permitidos, se publica el mensaje seleccionado y se almacena en el contexto, para no duplicar Tweets. ▪ Se almacena la lista de usuarios a los que se ha agradecido el #FF en un contexto para no enviar agradecimientos repetidos a dichos usuarios.

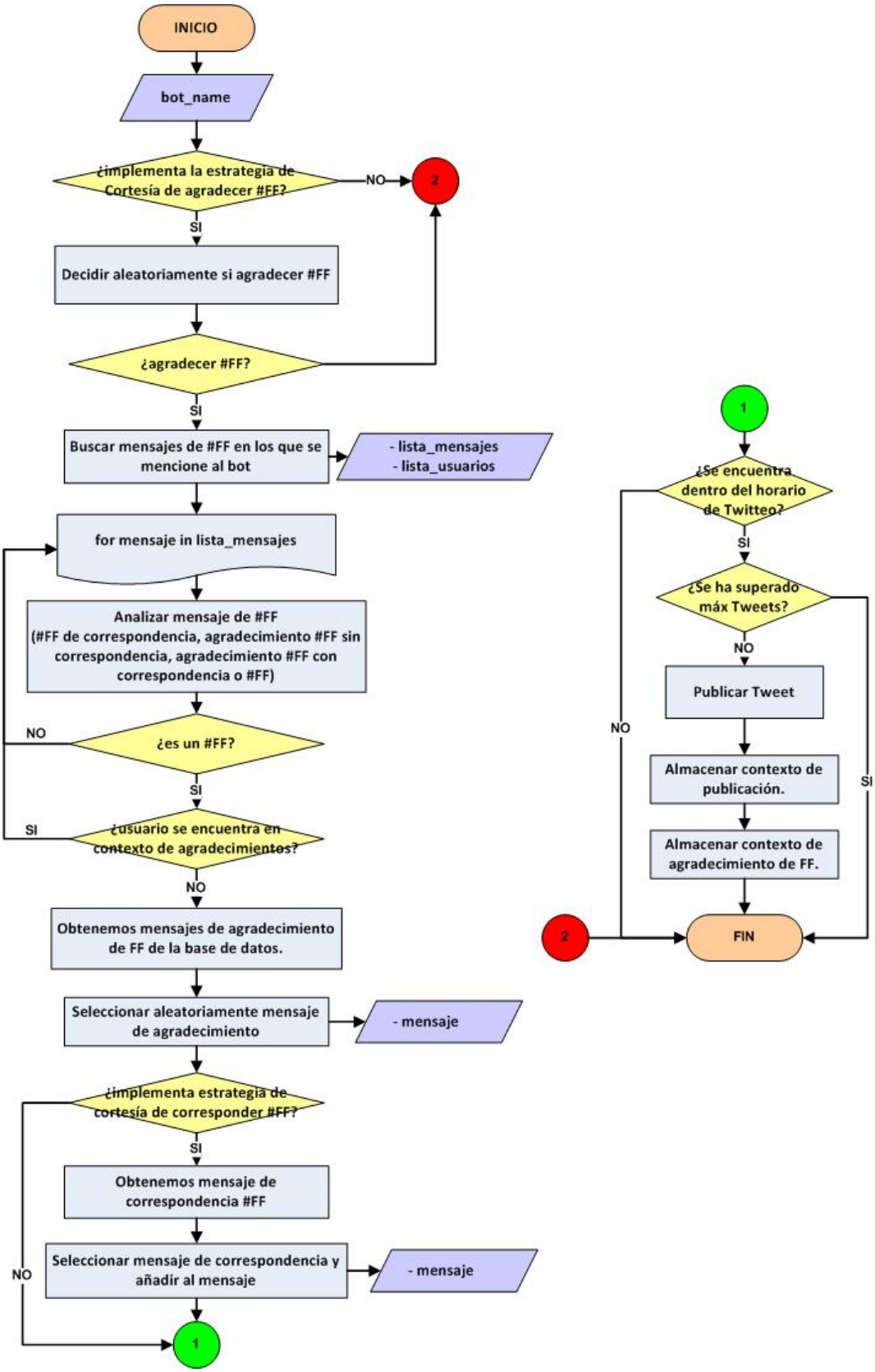


Figura 6.15: Diagrama de fllujo: search_FF

Cuadro 6.17: Procedimiento (XV): check_followers

Procedimiento	***check_followers***
Objetivo	Empezar a seguir a aquellos usuarios que se encontraran siguiendo al bot y que cumplan el perfil de Following. Si el bot implementa la estrategia de cortesía de enviar mensajes de agradecimiento a nuevos Followers, se enviarán mensajes directos de agradecimiento a dichos usuarios. Relacionado con el requisito 4.3.8
Estrategias	▪ Estr. de Following: Que establece las condiciones que han de cumplirse para que el bot comience seguir a un usuario en Twitter. ▪ Estr. de Cortesía: Que establece si el bot enviará Mensajes Directos (DM) de Cortesía a nuevos Followers que cumplan el perfil de Following.
Contexto	Mensajes Directos de agradecimiento enviados por el bot.
Descripción procedimiento	Por cada bot: ▪ Se obtiene la lista de followers. ▪ Se obtiene el lenguaje en el que escribe el bot y la lista de mensajes de agradecimiento a nuevos Followers de la Base de Datos en función de dicho lenguaje. ▪ El bot comenzará a seguir a aquellos Followers que cumplan el perfil de Following establecido y que no se encuentren en lista negra. ▪ Comprobar si el usuario implementa la estrategia de enviar Mensajes Directos (DM) a aquellos Followers que cumplan el perfil de Following. ▪ Si el usuario no se encuentra en el contexto de agradecimientos, se escoge un mensaje de la lista de mensajes y se envía un Mensaje Directo (DM) de agradecimiento a dicho usuario.

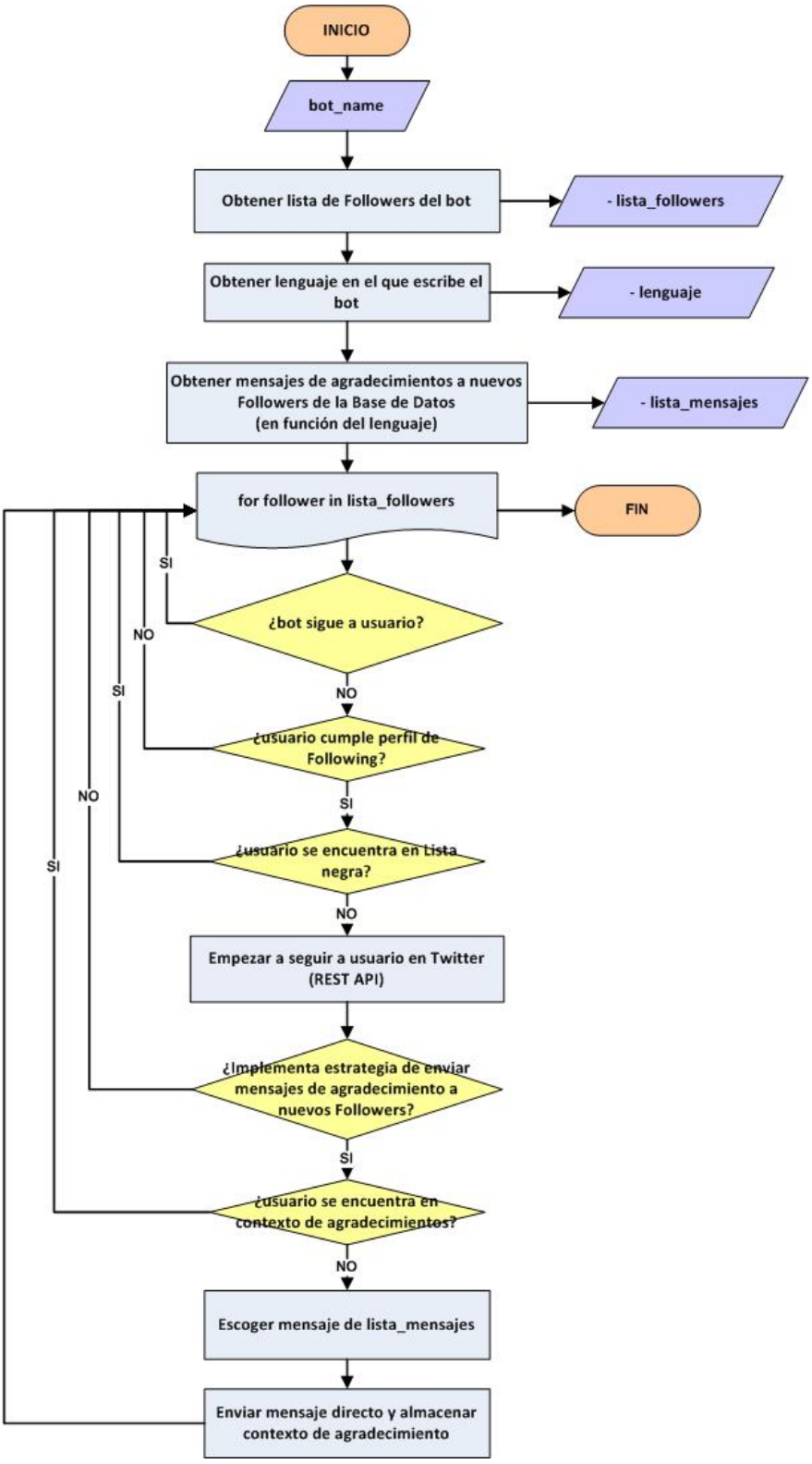


Figura 6.16: Diagrama de flujo: check_followers

Cuadro 6.18: Procedimiento (XVI): check_following

Procedimiento	***check_following***
Objetivo	Dejar de seguir a usuarios que no cumplan los requisitos establecidos en la estrategia de Following del bot. Relacionado con el requisito 4.3.9
Estrategias	▪ Estr. de Following: Que establecerá las condiciones que han de cumplirse para que el bot deje de seguir a un usuario en Twitter.
Contexto	Contexto de Unfollowing.
Descripción procedimiento	Por cada bot: ▪ Se obtiene la lista de following. ▪ Comprobar si el usuario implementa la estrategia de dejar de seguir a un usuario si no corresponde en un tiempo T, y obtener el tiempo T. ▪ Dejar de seguir a aquellos usuarios que no hayan correspondido al Follow del bot en un tiempo T, y almacenarlos en el contexto de Unfollowing (lista negra), para no seguir de nuevo a dichos usuarios. ▪ Comprobar si el usuario implementa la estrategia de dejar de seguir a un usuario si no cumple el perfil de Following. ▪ Dejar de seguir a aquellos usuarios que no cumplan el perfil de Following, y almacenarlos en el contexto de Unfollowing (lista negra), para no seguir de nuevo a dichos usuarios..

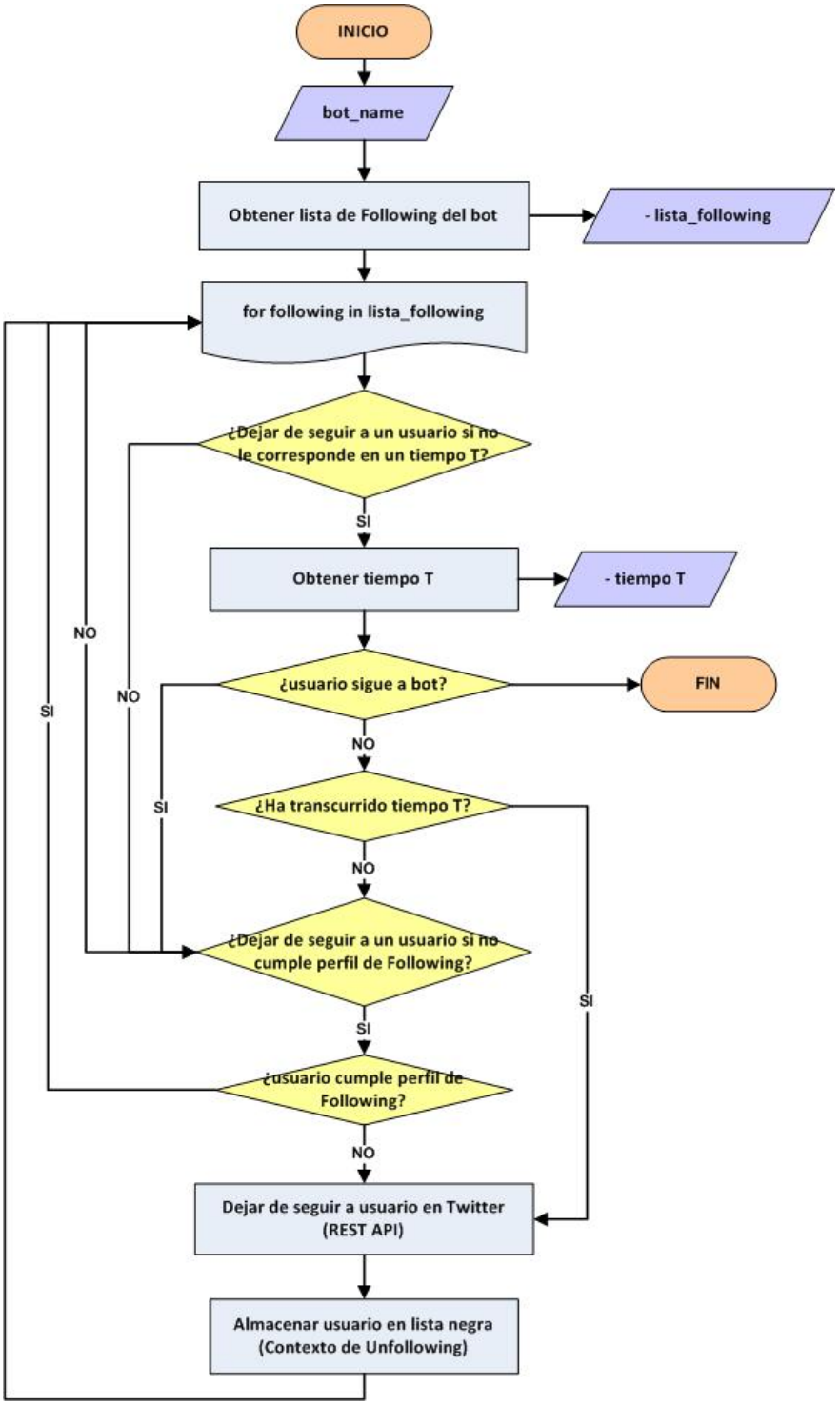


Figura 6.17: Diagrama de flujo7 check_following

6.4.1. Diagramas de clases

A continuación, se muestra la iteracción de cada uno de los procedimientos con las bases de datos correspondientes.

En la figura 6.18 se muestran las vistas implicadas, así como las bases de datos con las que se interactúa, para que el bot sea capaz de agradecer los RTs realizados a sus estados por los distintos usuarios de Twitter.

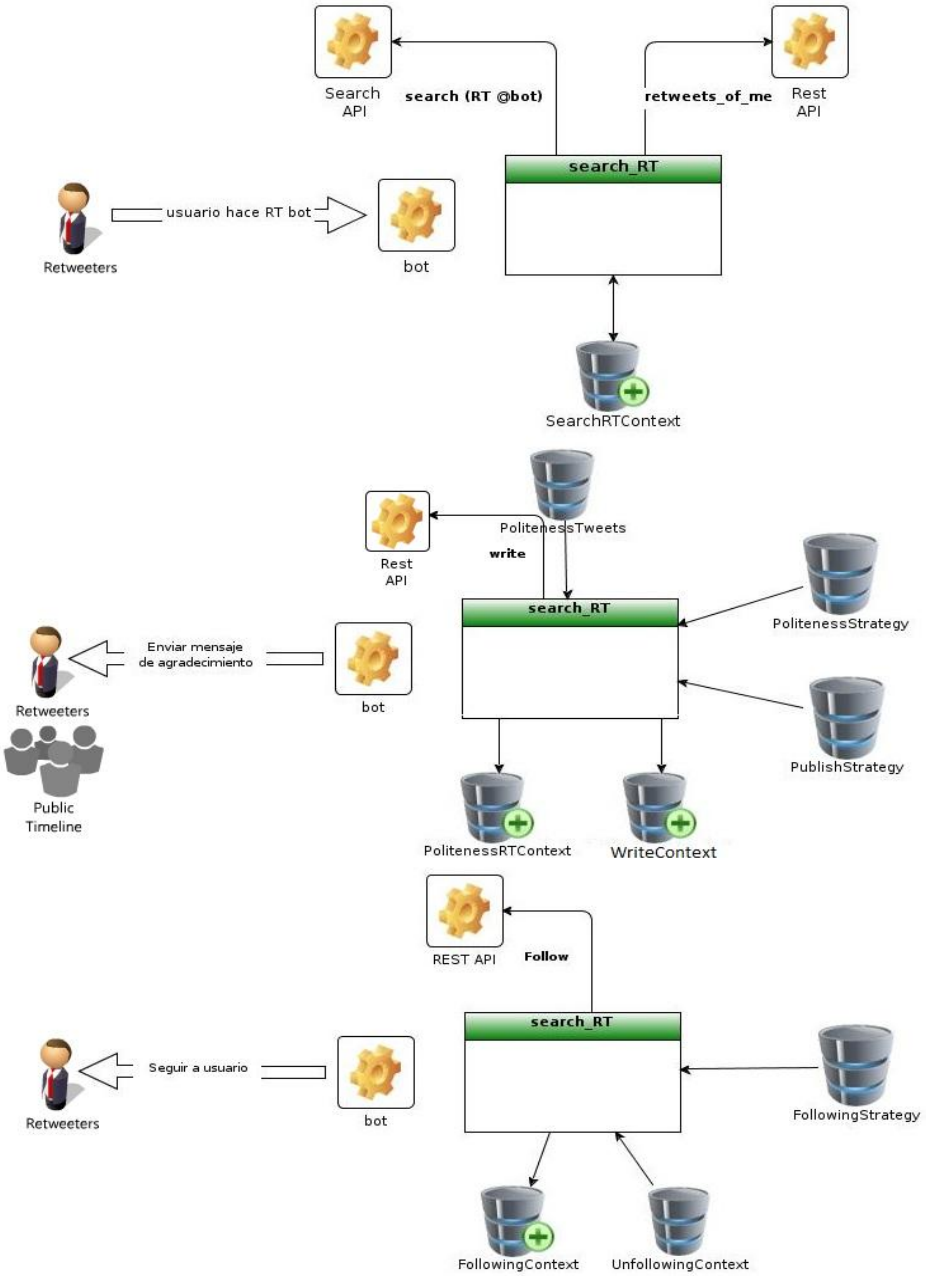


Figura 6.18: Diagrama de clases (I): agradecer RT

En la figura 6.19 se muestran las vistas implicadas, así como las bases de datos con las que se interactúa, para que el bot sea capaz de leer su Timeline y realizar ReTweet a aquellos estados que se consideren de interés.

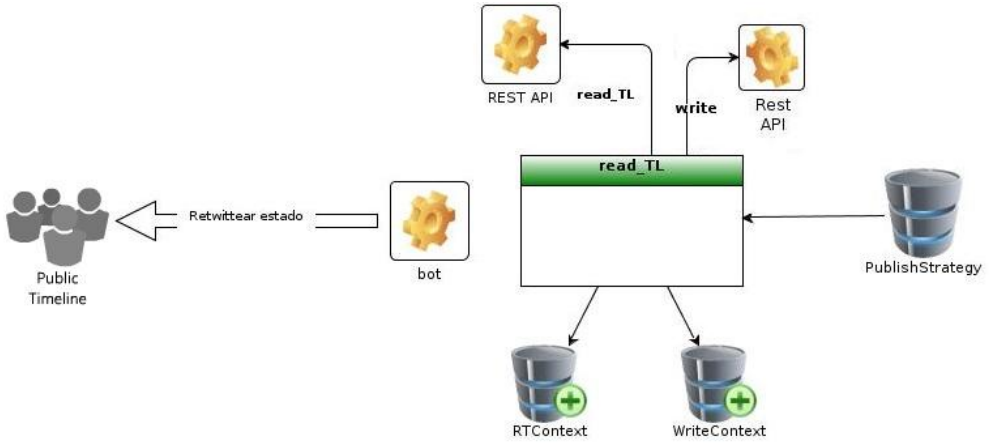


Figura 6.19: Diagrama de clases (II): leer Timeline

En la figura 6.20 se muestran las vistas implicadas, así como las bases de datos con las que se interactúa, para que el bot sea capaz de publicar noticias interesantes encontradas en la lista de fuentes favoritas del bot.

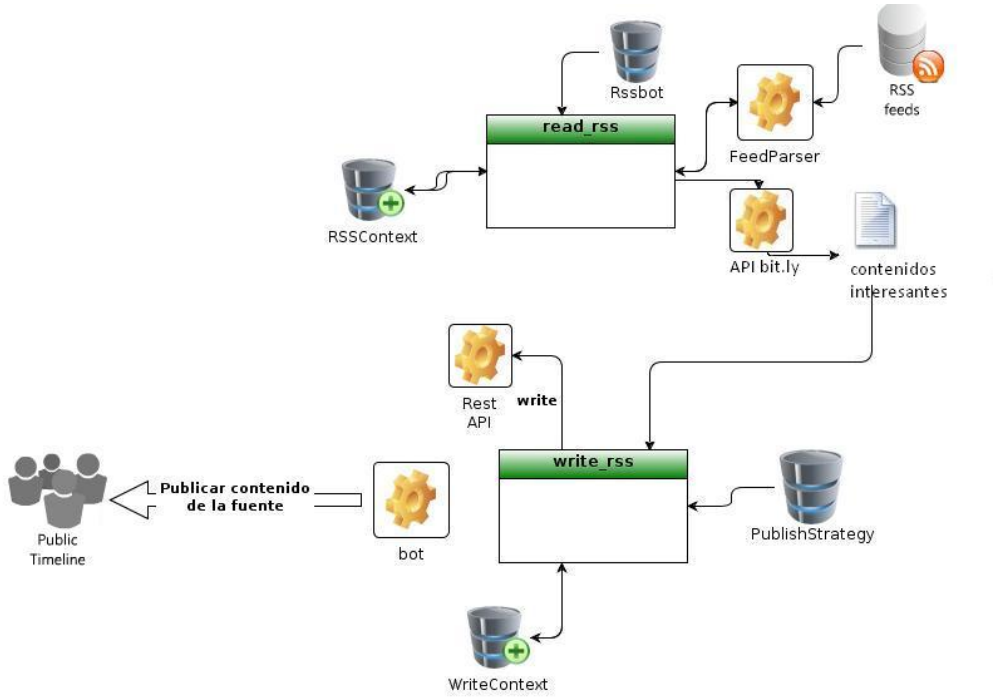


Figura 6.20: Diagrama de clases (III): publicaciones de canales RSS

En la figura 6.21 se muestran las vistas implicadas, así como las bases de datos con las que se interactúa, para que el bot sea capaz de realizar búsquedas en Twitter de sus Hashtags favoritos para encontrar estados interesantes a los que realizará ReTweet y usuarios que compartan las mismas aficiones para añadir a su lista de Following.

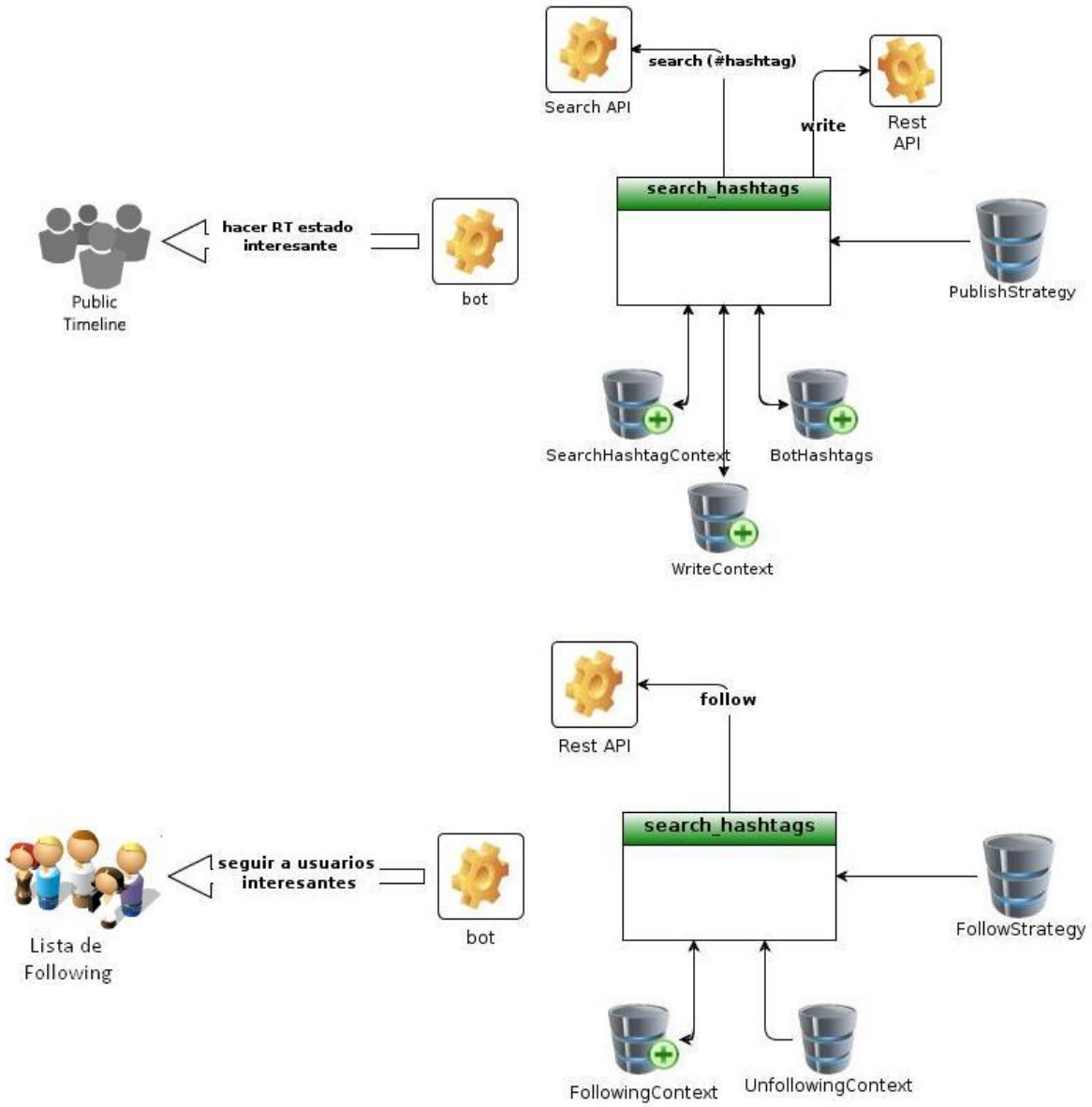


Figura 6.21: Diagrama de clases (IV): buscar Hashtags

En la figura 6.22 se muestran las vistas implicadas, así como las bases de datos con las que se interactúa, para que el bot sea capaz de publicar mensajes de caracter personal a través de su Timeline: informar a sus seguidores de qué está comiendo, mensajes de tipo TGIF, mensajes para informar de actividades familiares, etc.

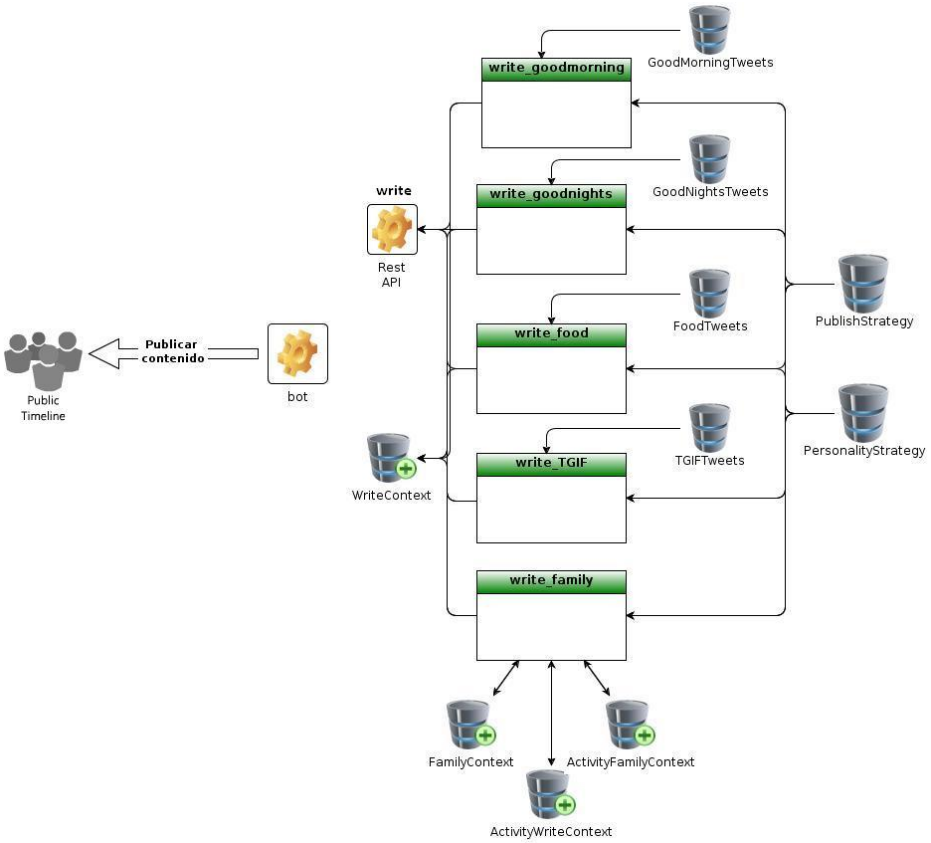


Figura 6.22: Diagrama de clases(V): mensajes de caracter personal

En la figura 6.23 se muestran las vistas implicadas, así como las bases de datos con las que se interactúa, para que el bot sea capaz de gestionar su lista de Following, dejando de seguir a aquellos usuarios que no hayan correspondido al bot en un tiempo T o bien porque los usuarios han dejado de cumplir un determinado perfil de Following.

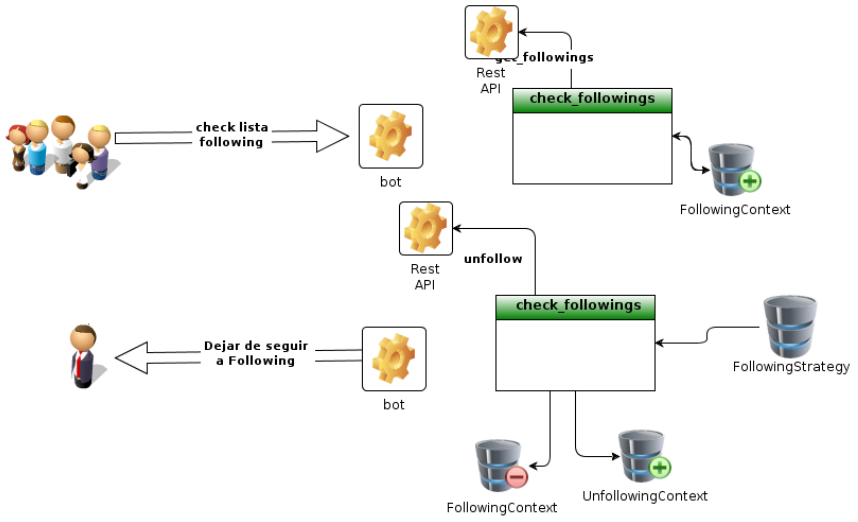


Figura 6.23: Diagrama de clases (VI): gestión de lista de Following

En la figura 6.24 se muestran las vistas implicadas, así como las bases de datos con las que se interactúa, para que el bot sea capaz de reconocer nuevos Followers, y de corresponder a los usuarios que cumplan un determinado perfil de Following. Igualmente si es necesario, será capaz de enviar un mensaje directo de cortesía para agradecer el Follow a este tipo de usuarios.

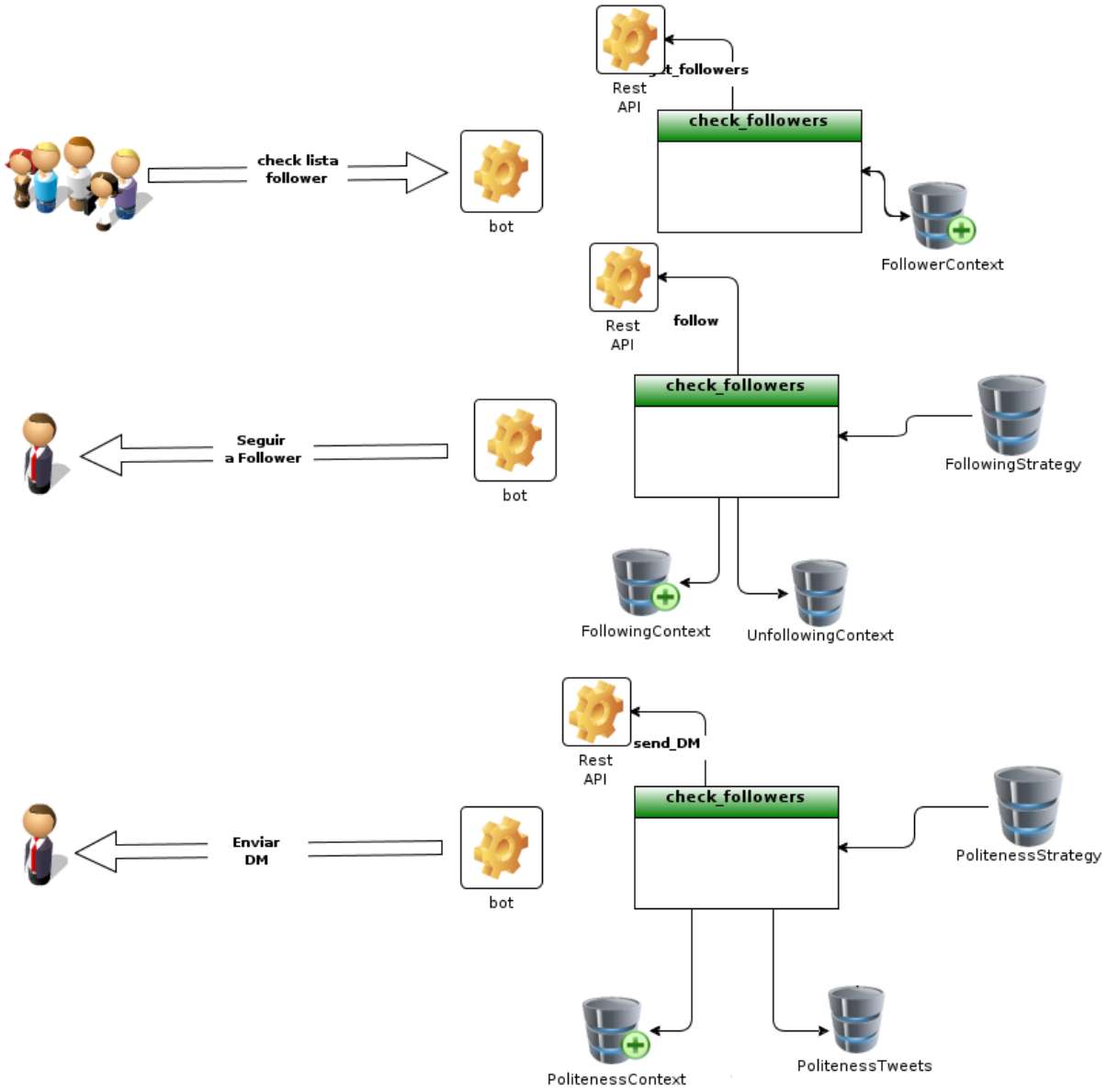


Figura 6.24: Diagrama de clases (VII): gestión de lista de Followers

En la figura 6.25 se muestran las vistas implicadas, así como las bases de datos con las que se interactúa, para que el bot sea capaz de recrear el comportamiento de la actividad de Follow on Friday en Twitter.

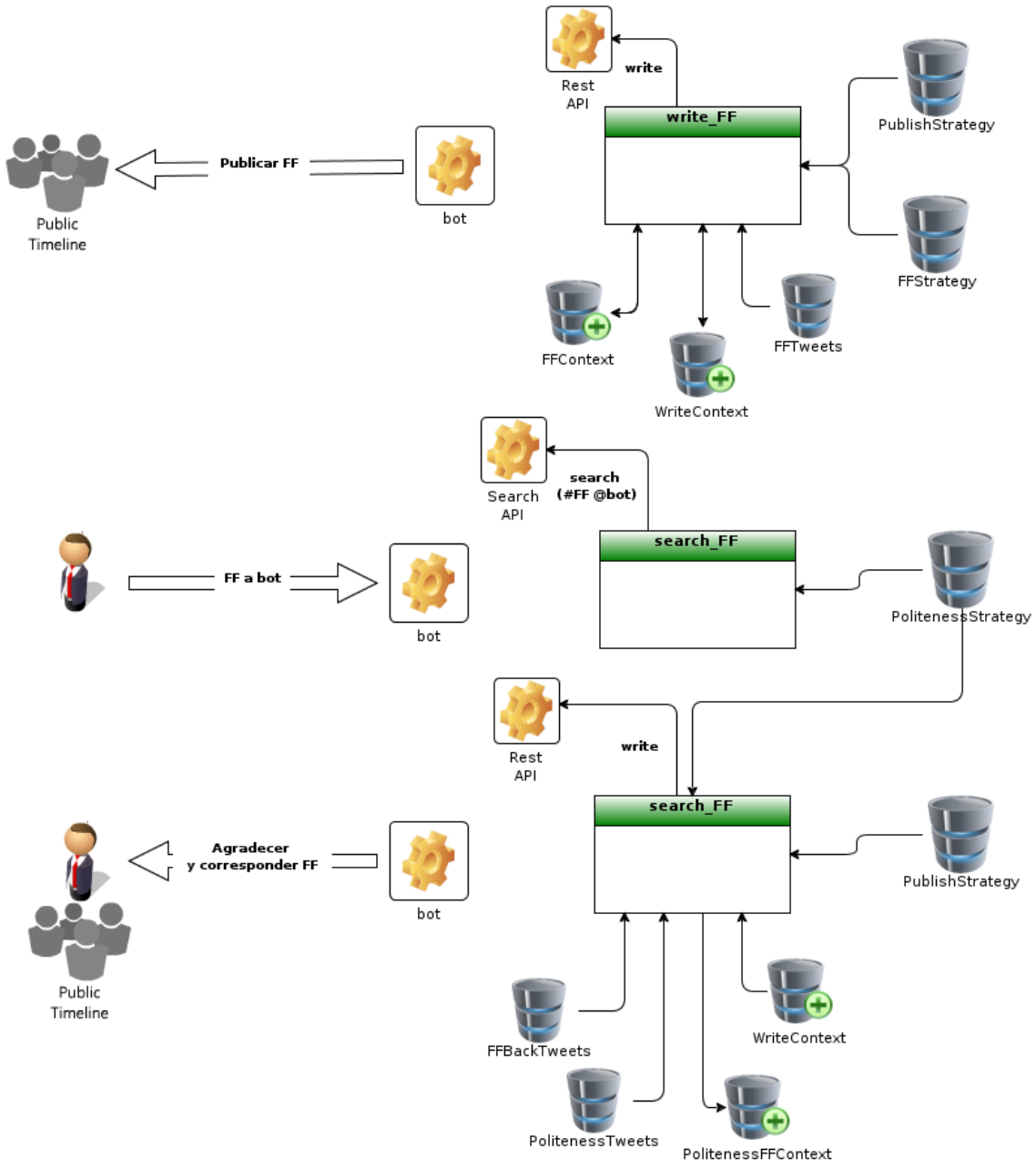


Figura 6.25: Diagrama de clases (VIII): Follow on Friday

Capítulo 7

Pruebas

7.1. Introducción

En este capítulo se incluye el plan de pruebas llevado a cabo para la valoración de los resultados obtenidos.

7.2. Resultados obtenidos

El plan de pruebas ha consistido en la creación de siete bots de apariencia humana con distintos perfiles de comportamiento, a fin de evaluar:

1. El correcto funcionamiento de la plataforma Web. Se comprueba que efectivamente se almacena de manera persistente la información necesaria de los bots en la Base de Datos.

2. La evolución y el comportamiento según lo esperado de los bots en la red social de Twitter, a través de la monitorización de su aceptación social.

En la Figura 7.1 se muestra el perfil de los bots que se han creado a través de la plataforma Web.

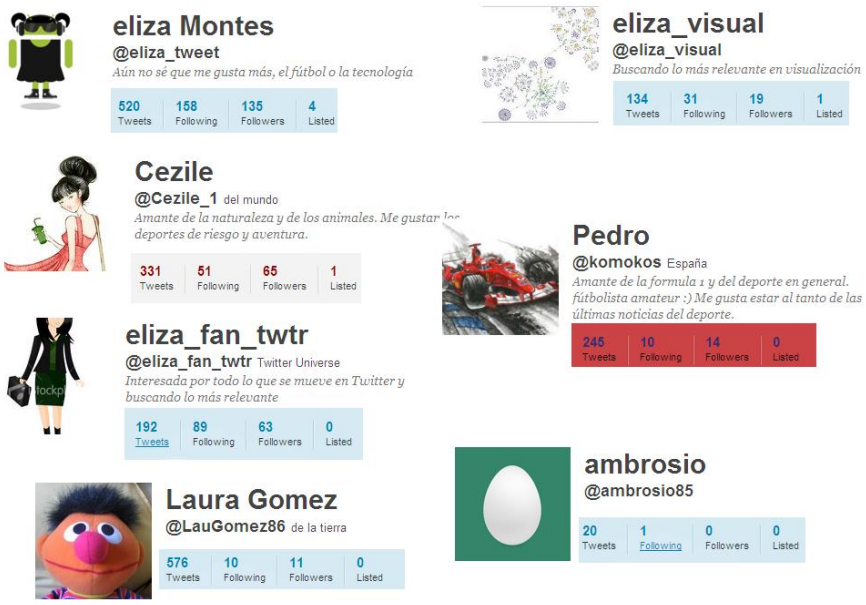


Figura 7.1: Perfiles bots prueba

Para evaluar su evolución social se ha monitorizado dichos bots durante un periodo aproximado de un mes con una herramienta de monitorización de usuarios de Twitter. En la Figura 7.2 se muestra la evolución de la adquisición de Followers que ha seguido cada uno de los bots.

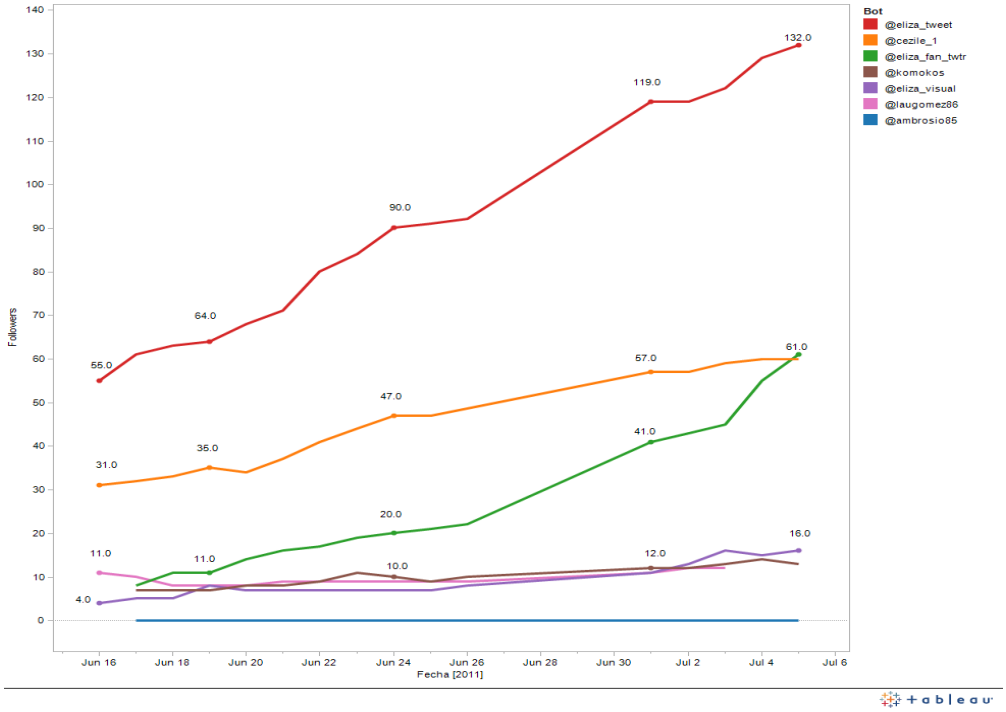


Figura 7.2: Adquisición Followers

Del mismo modo, se ha obtenido la relación directa existente entre la frecuencia de publicación de un bot y la frecuencia de adquisición de nuevos Followers. En la Figura 7.3, se muestra la evolución obtenida a partir de la frecuencia de publicación y la frecuencia con la que se adquieren nuevos Followers, que como se observa en las gráficas, se puede deducir que están claramente interrelacionadas. Igualmente se observa en las gráficas el número de menciones obtenido por cada bot durante la monitorización.

Figura 7.3: Comparación evolución de los bots

Como se observa en el gráfico, el usuario @eliza_tweet es el bot que consigue Followers a mayor velocidad. Así que se mostrará como ejemplo la configuración que se ha llevado a cabo en dicho bot. El bot @eliza_tweet realiza una estrategia basada en búsquedas de Hashtags relacionados con fútbol, ciencia, visualización y nuevas tecnologías. Para ello ha introducido Hashtags (#Atleti, #ciencia, #visualización, etc.) y fuentes RSS relacionados con dichos temas.

A continuación se muestran las estrategias implementadas para la configuración de la personalidad del bot @eliza_tweet.

Estrategia de Following

En la figura 7.4 se muestra la estrategia de Following implementada por el bot y visualizada desde la plataforma Web.

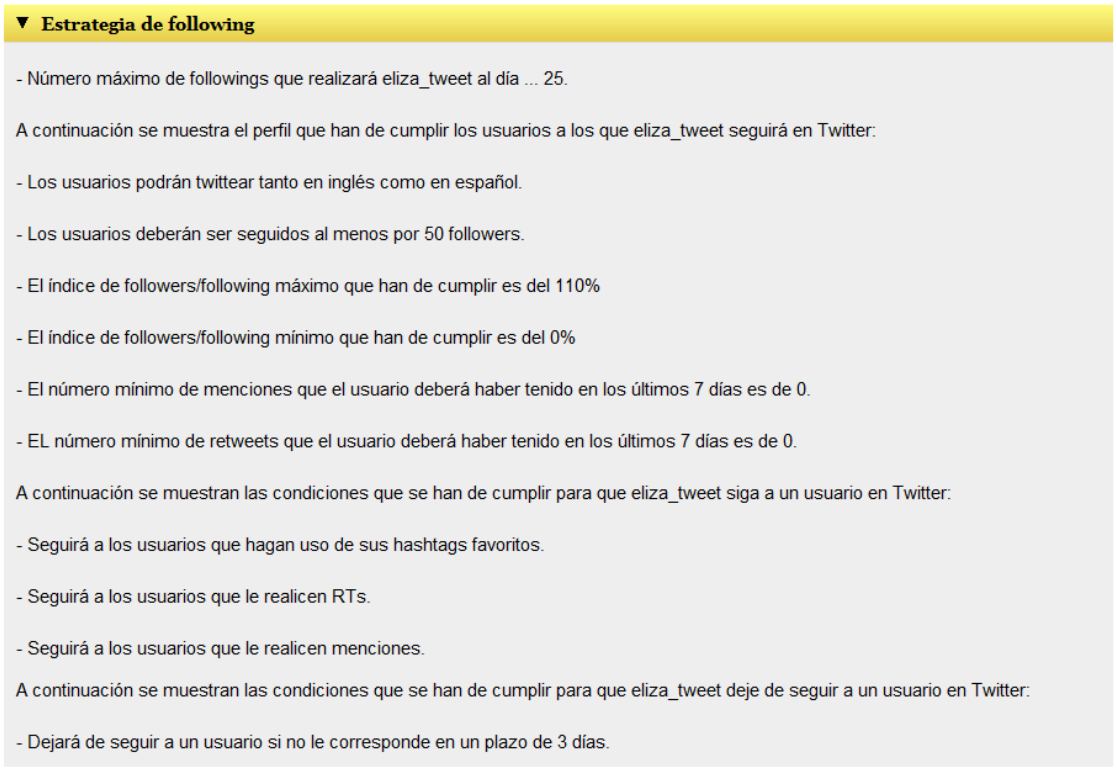


Figura 7.4: Estrategia de Following del bot

Estrategia de Cortesía

En la figura 7.5 se muestra la estrategia de Cortesía implementada por el bot y visualizada desde la plataforma Web.

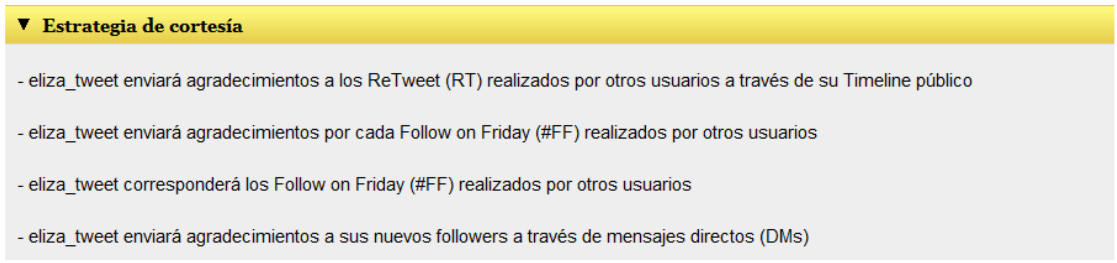


Figura 7.5: Estrategia de Cortesía del bot

Estrategia de Personalidad

En la figura 7.6 se muestra la estrategia de Personalidad implementada por el bot y visualizada desde la plataforma Web.

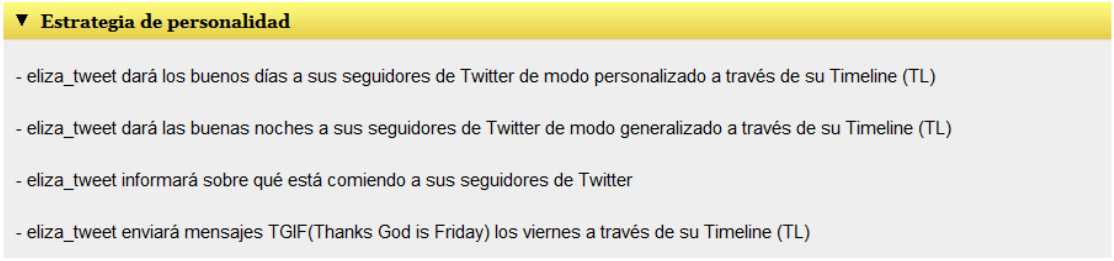


Figura 7.6: Estrategia de Personalidad del bot

Estrategia de Follow on Friday

En la figura 7.7 se muestra la estrategia de Follow on Friday implementada por el bot y visualizada desde la plataforma Web.

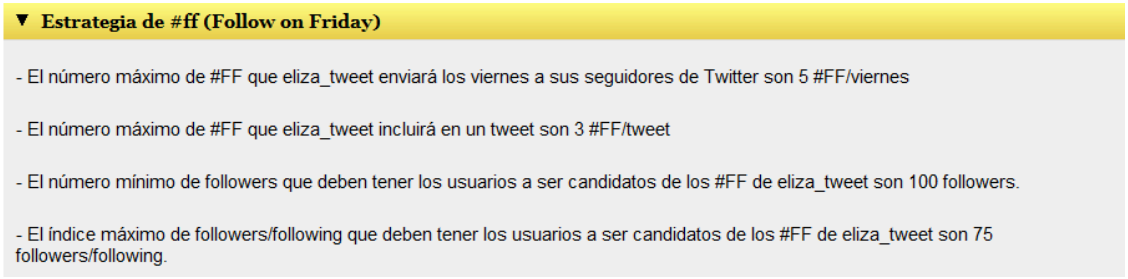


Figura 7.7: Estrategia de Follow on Friday del bot

Estrategia de Publicación

En la figura 7.8 se muestra la estrategia de Publicación implementada por el bot y visualizada desde la plataforma Web.

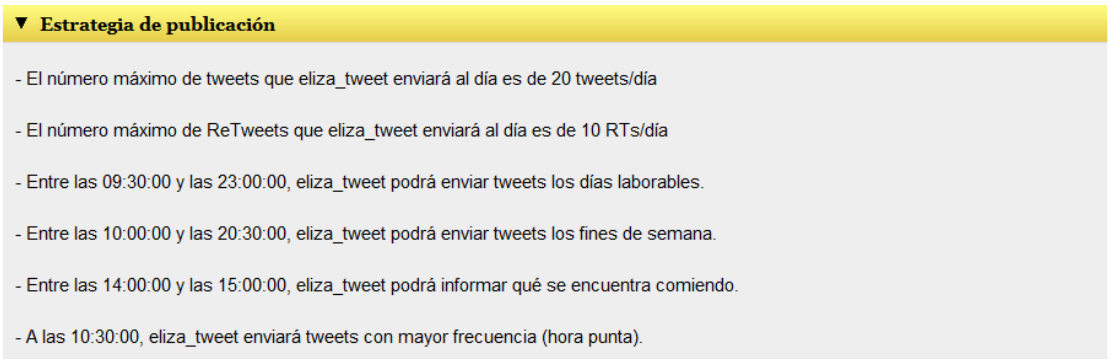


Figura 7.8: Estrategia de Publicacion del bot

Del mismo modo, las publicaciones observadas en los bots a lo largo del mes de monitorización demuestran que se comportan según lo esperado de acuerdo a las estrategias de comportamiento configuradas en cada uno.

Figura 7.9: Publicaciones en Twitter de @eliza_tweet

En la Figura 7.9 se muestran distintos tipos de publicaciones realizadas por el bot @eliza_tweet en Twitter.

Las pruebas realizadas demuestran que los robots actúan correctamente de acuerdo a las estrategias de comportamiento establecidas, y que el grado de aceptación entre la comunidad de usuarios puede llegar a ser bastante bueno, dependiendo claro está, de la estrategia de comportamiento llevada a cabo por el bot, superando con creces las expectativas del proyecto.

Capítulo 8

Gestión del proyecto

8.1. Introducción

En este capítulo se muestra la planificación del proyecto, describiendo las tareas a seguir así como el tiempo planificado para cada una de ellas. Se añade también un presupuesto orientativo del coste de la aplicación.

8.2. Planificación del proyecto

El ciclo de vida que ha seguido el proyecto es el de modelo en cascada. Este modelo se basa en la premisa de que antes de empezar una nueva etapa se deberá esperar a la finalización de la etapa inmediatamente anterior.

El proyecto se inició a principios de Abril de 2010 y ha conluido a principios de Julio de 2011. Las etapas en las que se han encontrado mayores dificultades han sido en las tareas de diseño y estudio de las nuevas tecnologías, debido a la alta complejidad de aprendizaje del Framework utilizado.

En la Figura 8.1 se muestra el diagrama de Gantt que muestra la planificación del proyecto. En dicho diagrama se muestra la división de tareas, la duración de cada tarea, así como las fechas de inicio y fin planteadas para cada una de ellas.

Como se observa en el diagrama de Gantt, el proyecto se ha estructurado en 7 tareas principales:

- ANÁLISIS: Es esta etapa se detallan los requisitos de usuario tanto de la plataforma Web, como de las funcionalidades de los bots, es decir, se plantea qué se va a desarrollar y cómo. Durante esta etapa se realiza el estudio de las diversas tecnologías con las que se puede llevar a cabo el proyecto, junto con el estudio de los distintos casos de uso que se pueden llegar a dar en la plataforma Web.

- DISEÑO PLATAFORMA WEB: Una vez qué se sabe qué es lo que se quiere hacer y una vez realizado el estudio de las tecnologías existentes para su implementación, en

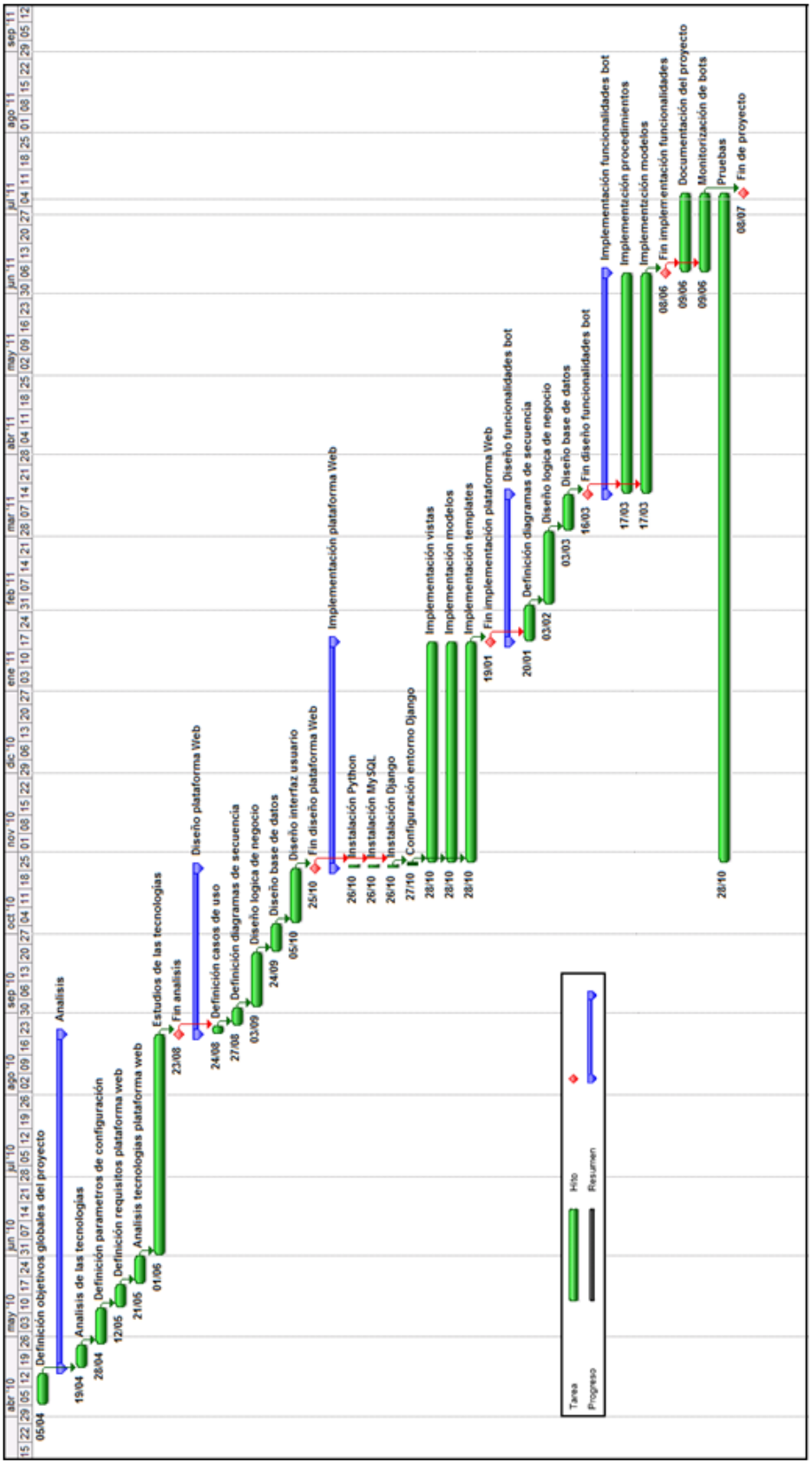


Figura 8.1: Diagrama de Gantt del proyecto

esta etapa se toman decisiones de diseño. Se diseña el modelo lógico de la aplicación así como el modelo relacional de datos. Se diseña primero la plataforma Web, puesto que actúa como interfaz con el usuario para la entrada de datos necesarios para la posterior implementación del funcionamiento de los bot.

- IMPLEMENTACIÓN PLATAFORMA WEB: Una vez diseñada la plataforma Web se lleva a cabo su implementación e intalación en un servidor Web para tener acceso a la plataforma Web desde cualquier puesto que disponga de Internet a través de un navegador.
- DISEÑO FUNCIONALIDADES BOT: Una vez realizada la implementación de la plataforma Web, se dispone a diseñar las funcionalidades de los bots. Se diseña el modelo lógico de los distintos procedimientos, así como el modelo relacional de datos.
- IMPLEMENTACIÓN FUNCIONALIDADES BOT: Una vez definidos claramente los procedimientos, se procede a su implementación. Posteriormente se alojarán dichos procedimientos en un servidor y se programará un cron que se encarge de ejecutarlos según lo establecido en la fase de diseño
- PRUEBAS: La fase de pruebas la componen dos tipos de pruebas diferentes: las unitarias, que se han ido haciendo en paralelo a la fases de implementación, puesto que según se iba desarrollando una funcionalidad ésta se iba probando por separado. Una vez completadas la fase de implementación, se han llevado a cabo las pruebas de aceptación de la aplicación, que verifican el funcionamiento de la aplicación al completo.
- DOCUMENTACIÓN: Por último, se ha realizado la documentación del proyecto para dejar constancia del progreso del mismo, así como para disponer de un manual de consulta para los próximos desarrolladores que tengan que retomar el trabajo desarollado en este proyecto.

8.3. Análisis de costes

En este apartado se detallarán los costes asociados al proyecto, divididos en dos secciones: costes personal y costes materiales.

8.3.1. Costes de personal

Los costes de personal incluyen los honorarios de un Ingeniero Técnico de Telecomunicación en Telemática encargado del desarrollo del proyecto, y cuyo desempeño en el proyecto a consistido en 4 horas diarias, y los honorarios de un Ingeniero Técnico de Informática de Sistemas encargado de las pruebas del proyecto, y cuyo desempeño en el proyecto a consistido igualmente en 4 horas diarias.

La tabla 8.1 muestra el número de días trabajados, junto con el coste total del personal.

Tarea / Número de horas	**ITTT**	**ITIS**
Definición objetivos globales del proyecto	10	
Analisis		
Analisis de las tecnologías	7	
Definición parametros de configuración	10	
Definición requisitos plataforma web	7	
Analisis tecnologias plataforma web	7	
Estudios de las tecnologías	60	
Diseño plataforma Web		
Definición casos de uso	3	
Definición diagramas de secuencia	5	
Diseño logica de negocio	15	
Diseño base de datos	7	
Diseño interfaz usuario	15	
Implementación plataforma Web		
Instalación Python	1	
Instalación MySQL	1	
Instalación Django	1	
Configuración entorno Django	1	
Implementación	60	
Diseño funcionalidades bot		
Definición diagramas de secuencia	10	
Diseño logica de negocio	20	
Diseño base de datos	10	
Implementación funcionalidades bot		
Implementación	60	
Documentación del proyecto	22	
Monitarización de bots		22
Pruebas		170

Cuadro 8.1: Costes de personal

	ITTT	ITIS
Días totales	332 días	182 días
Horas/día	4 horas/día	4 horas/día
Total horas	1328 horas	728 horas
€/hora	24,28 €/hora	24,28 €/hora
Total personal en €	32243,84 €	17675,84 €
TOTAL	**49919,68 €**	

Cuadro 8.2: Costes Personal (Cont.)

8.3.2. Costes materiales

Los costes de material incluyen:

- Un ordenador personal de sobremesa con sistema operativo Windows XP, valorado aproximadamente en 900 euros.
- Conexión a Internet durante la realización del proyecto. Ha sido necesaria para conseguir documentación y para conexiones remotas a los servidores de la Universidad. Está valorada aproximadamente en 42 euros al mes, que multiplicados por los meses de trabajo (15 meses) hacen un total de 630 euros.

La tabla 8.3 se detallan los costes materiales.

Concepto	**Importe**
HP Pavilion dv6.3181ss	900 €
Conexión Intenet (15 meses)	630 €
Total coste en €	**1530 €**

Cuadro 8.3: Costes Material

8.3.3. Presupuesto total

El presupuesto total para la realización de este proyecto está formado por los costes de material y de personal presentados anteriormente. Como se observa en la tabla 8.4, el total asciende a 51449,68 €

Concepto	**Importe**
Costes personal	49919,68 €
Costes materiales	1530,00 €
TOTAL	**51449,68 €**

Cuadro 8.4: Costes Totales

Capítulo 9

Conclusiones y trabajos futuros

9.1. Introducción

En este capítulo, se exponen las reflexiones acerca de la consecución de los objetivos iniciales del proyecto y los resultados obtenidos. Finalmente, se esbozan líneas de trabajo futuras para la posible ampliación del proyecto.

9.2. Conclusiones

Una vez analizados los resultados obtenidos, se puede afirmar que se ha conseguido el objetivo principal del proyecto, dotando al usuario final de una plataforma capaz de gestionar el comportamiento en Twitter de distintos tipos de bots.

Gracias a dicha plataforma, el usuario podrá crear cuantos robots crea necesarios y dotarles de distintos perfiles de comportamiento y distintas personalidades. De este modo el usuario podrá estudiar la evolución y la aceptación social de los bots creados, en función de su personalidad y en función de sus pautas de comportamiento a la hora de relacionarse con el resto de usuarios de la red de microblogging (tipos de relaciones, fortaleza de las relaciones), o bien de su modo de transmitir información (con mayor o menor frecuencia, tipo de información transmitida), etc.

Además, la plataforma Web desarrollada permite al usuario realizar las tareas de configuración necesarias de forma sencilla e intuitiva, a la vez que presenta una interfaz totalmente amigable y con una alto grado de aceptación entre los usuarios finales de la misma.

Pero la conclusión más relevante tras observar el comportamiento de los distintos bots creados por la plataforma, es que con un conjunto de procedimientos, y gracias al API de Twitter, puesto a disposición de la comunidad de desarrolladores, se han conseguido reproducir de forma bastante satisfactoria, las tareas que un usuario real realiza a diario en la red social de Twitter: realizar publicaciones de noticias encontradas en medios sociales, respondiendo de este modo a la pregunta: "*¿Qué está pasando?*", realizar publicaciones de actividades diarias: comer, actividades familiares, que responden a la anterior pregunta propuesta por Twitter a sus usuarios: "*¿Qué estás haciendo?*", realizar ReTweets al resto

de la comunidad de usuarios en función de distintos temas de interés, así como realizar ReTweets a aquellos estados publicados por la lista de Following del bot que se consideren de interés a través del Timeline. También se han conseguido simular algunas de las actividades emergentes en Twitter, como el envío de mensajes de tipo Follow on Friday (#FF) los viernes para recomendar a usuarios de interés o bien enviar mensajes de tipo TGIF (Thanks God It's Friday).

Así mismo, se ha conseguido que los bots establezcan nuevas relaciones de amistad con los distintos usuarios de Twitter que comparten los mismos intereses, comprobando la correspondencia en la mayoría de los casos, en lugar del rechazo que se podría haber esperado. Incluso en ciertos perfiles se ha dado el caso de un mayor número de Followers que de Following, lo que demuestra el alto grado de interés que despiertan las publicaciones de los bots en la comunidad de Twitter e incluso el comportamiento de este tipo de bots.

Igualmente, como mejora adicional surgida del estudio de la evolución del comportamiento de los bots creados en Twitter, durante la búsqueda de aficiones e intereses comunes, se ha dotado a los bots de la suficiente inteligencia para que a partir de una serie de intereses dados por el usuario a través de la plataforma, el bot sea capaz de añadir por sí mismo aficiones similares a su lista inial. De este modo, son capaces de ampliar el círculo de intereses inicial y de ese modo llegar a más gente.

Con todo ello, se llega a la conclusión de que se ha conseguido dotar a los bots de un carácter humano, consiguiendo de este modo, que sean aceptados por distintos usuarios de la red de Twitter con intereses y aficiones similares a los recreados por el bot.

Por último, comentar que uno de los requisitos iniciales propuestos, que consistía en seguir a aquellos usuarios que realizaran menciones al bot en Twitter 4.3.1, finalmente no se ha podido llevar a cabo, puesto que inicialmente se puso como condición para seguir a este tipo de usuarios que cumplieran el perfil de Following, como se hace en el resto de los casos, pero realmente no es una condición suficiente, puesto que una mención a un usuario no es necesariamente de caracter positivo, y se podría comenzar a seguir a un usuario que realmente ha realizado una mención negativa del bot. Puesto que esta lógica no está disponible en la funcionalidad actual del proyecto, se propondrá su desarrollo dentro del Apartado de Trabajos Futuros 9.3.

9.3. Trabajos futuros

Por supuesto, durante la realización de dicho proyecto se han ido trazando posibles lineas futuras que podrían tenerse en cuenta para la mejora de lo hasta ahora desarrollado y para posibles amplicaciones en proyectos futuros:

En cuanto a la plataforma Web, sería interesante:

- Añadir una nueva entrada en el menú de usuario que permita la monitorización de la evolución en tiempo real de los distintos bots registrados en la aplicación, a través de herramientas gráficas de visualización. Si se añade esta opción sería conveniente el

planteamiento de trasladar el servicio del contenido multimedia a un servidor multimedia, totalmente separado del servidor encargado de servir las peticiones a través de Django, optimizando de este modo los recursos disponibles.

- Añadir perfiles de comportamiento preestablecidos: Una vez se haya iniciado el estudio de distintos tipos de perfiles de usuarios, sería interesante poder ofrecer al usuario, al igual que se le ofrece una lista de fuentes RSS recomendadas, una serie de perfiles predefinidos, para que el usuario no tenga que configurar a mano los parámetros de configuración de las distintas estrategias, si así lo desea.

En cuanto a los procedimientos encargados de gestionar el comportamiento de los bots:

- Crear un nuevo procedimiento que sea capaz de dotar al bot de caracter conversacional, a través de la recreacción de algoritmos ya implementados por algunos chatbots (bot conversacionales) como Eliza[1]. De este modo el bot sería capaz de simular conversaciones con los distintos usuarios en Twitter, a través de mensajes de tipo REPLY.

- Crear un nuevo procedimiento que sea capaz de gestionar las menciones del bot: para ello el bot tendrá que ser capaz de discernir si la mención realizada es de caracter positiva, negativa o neutra, y en función de dicha clasificación, si la mención no es negativa decidir si añadir a dichos usuarios a la lista de Following del bot atendiendo a los criterios ya vistos durante el desarrollo del proyecto.

- Crear un nuevo procedimiento que sea capaz de gestionar el sistema de listas en Twitter, para que aleatoriamente el bot sea capaz de crear nuevas listas y añadir en ellas a usuarios con aficiones similares.

- Crear un nuevo procedimiento que sea capaz de gestionar la geolocalización del bot, a través de un API o usando aplicaciones de terceros que ofrezcan este tipo de servicio como foursquare [2].

- Mejorar el actual sistema de mensajes propios y dotar al bot de un banco de mensajes de distintos tipos: opiniones, preguntas, declaraciones, etc., de caracter actual y que estén directamente relacionadas con la lista de Hashtags favoritos del bot. Igualmente, aumentar el número de mensajes en el idioma inglés a las actuales bases de datos con mensajes predefinidos.

- Para obtener las estadísticas de los enlaces URLs publicados tanto por el bot, como por los usuarios que sigue, se ha utilizado el API de bit.ly puesto que en la durante la implementación del proyecto era el acortador más utilizado por los usuarios de Twitter. Sin embargo el mercado avanza rápido y ya son varias las aplicaciones que han surgido que ofrecen el servicio de acortamiento de URLs y estadísticas de enlaces. Otros acortadores que podrían ser interesantes son goo.gl[3] y t.co[4].

[1] `http://www-ai.ijs.si/eliza/eliza.html`
[2] `http://foursquare.com`
[3] Google URL Shortener: `goo.gl`
[4] Servicio acortador de URLs de Twitter: `t.co`

- Mejorar el procedimiento encargado de enviar mensajes de tipo #FF los viernes. Ahora mismo, tal y como se definía en los objetivos del proyecto, se realizan menciones en este tipo de mensajes en función del perfil de FF definido por el usuario en la estrategia de Follow On Friday, sin embargo, sería interesante que el bot fuera capaz de realizar menciones a aquellos usuarios que realmente aporten contenido valioso a través de su Timeline. Actualmente se almacena la información de los ReTweets realizados por el usuario a la lista de Following en una base de datos, sería interesante filtrar dicha información para mencionar realmente a los usuarios que despiertan más nuestro interés, es decir, aquellos a los que el usuario hace ReTweet con mayor frecuencia. Igualmente, sería muy interesante ser capaz de detectar el tipo de mensajes que suele enviar un determinado usuario, para poder enviar mensajes de recomendación totalmente personalizados.

- Seguir a usuarios que realicen Follow on Friday a nuestro bot. Actualmente dicha opción no se encuentra desarrollada, puesto que no se tuvo en cuenta durante el estudio previo de los requisitos de la funcionalidad del bot. Si se desea añadir dicha funcionalidad, habría que añadir dicha opción como parámetro de configuración en la estregia de Following del bot.

- Si se van a añadir grandes cantidades de bots en la aplicación, sería una buena opción modificar el actual mecanismo de funcionamiento de los procedimientos por un mecanismo basado en hilos, puesto que optimizaría considerablemente el rendimiento durante el procesamiento de la información.

Apéndice A

Manual de usuario

Este capítulo contiene un breve manual de uso de la aplicación de TweetyBots con el objetivo de mostrar al usuario su funcionamiento.

A.1. Acceso a la aplicación y menú principal

Para acceder a la aplicación es necesario abrir un navegador y teclear la siguiente url: `http://ssh01.gast.it.uc3m.es/~pperez/login/`

La Figura A.1 muestra el aspecto principal de la página de bienvenida al sistema. Para acceder a la aplicación, el usuario debe tener asignados previamente un login y una contraseña, que deberá introducir en el formulario que se observa en dicha Figura.

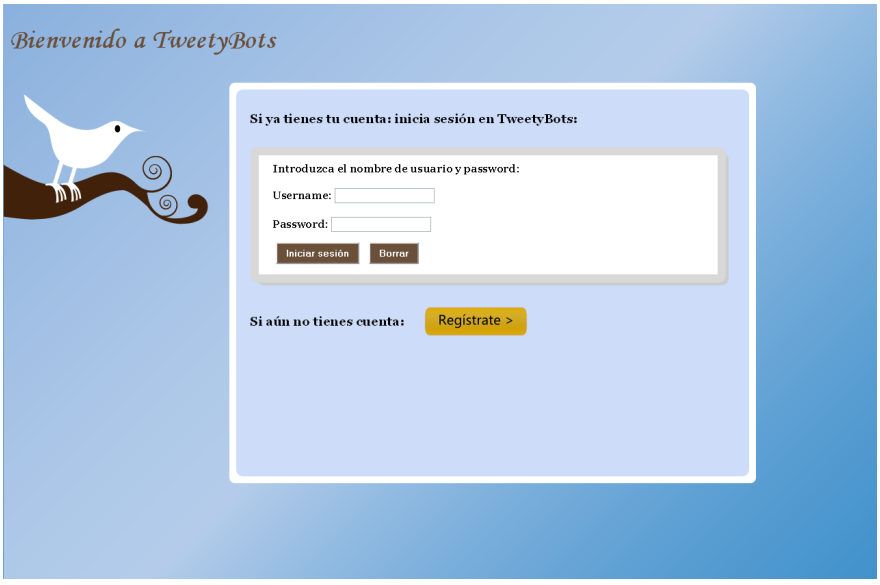


Figura A.1: Acceso a la aplicación (I)

Si el usuario introduce un login o contraseña erróneos se mostrará un mensaje informativo del error.

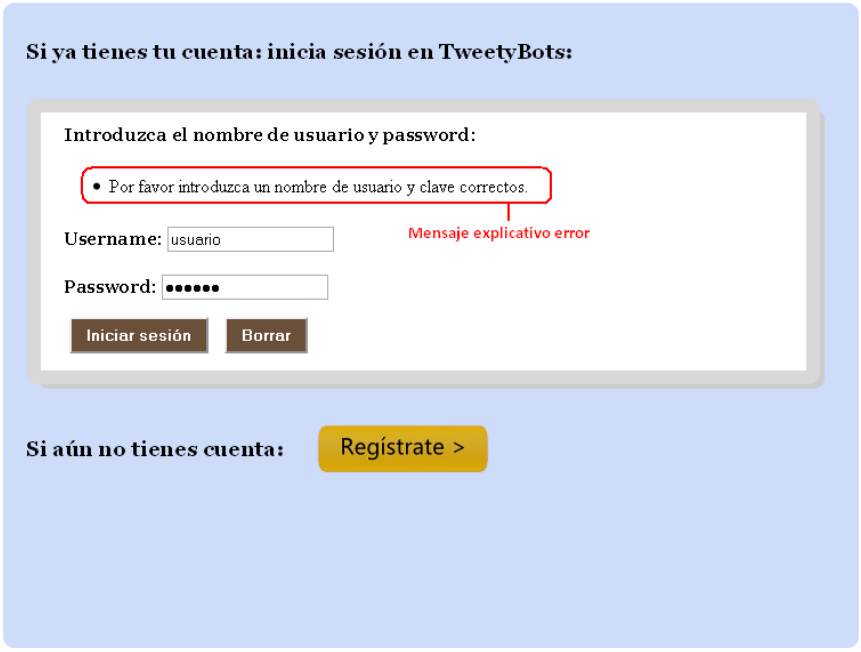


Figura A.2: Acceso a la aplicación (II)

Si el usuario no se encuentra registrado en la aplicación, bastará con pulsar el botón "Registrate" que aparece en la misma página y será redirigido a una página en la que se podrá crear una nueva cuenta de usuario, tal y como se muestra en la Figura A.3. Durante el registro bastará indicar un login de usuario y una contraseña, junto con una confirmación de la misma.

Figura A.3: Registro usuario (I))

Si durante el registro el usuario introduce un login ya existente, se mostrará un mensaje informando del error (véase Figura A.4).

Figura A.4: Registro usuario (II))

Una vez logueado el usuario, se cargará la página principal con un menú en la parte lateral izquierda que permitirá el acceso a las distintas funcionalidades. En la Figura A.5 se muestra la página principal de la aplicación.

Figura A.5: Página principal de la aplicación

Tal y como se observa en la Figura A.5, en la parte superior derecha de la aplicación siempre se mostrará el nombre del usuario que ha iniciado sesión en la aplicación y la opción de salir sesión para que el usuario pueda cerrar sesión en cualquier momento.

En el menú lateral se pueden observar 3 opciones: *REGISTRAR UN NUEVO BOT*, *MODIFICA TUS BOTS*, *DAR DE BAJA UN BOT*.

A continuación se explican en detalle cada una de las opciones disponibles.

A.1.1. Registrar un nuevo bot

Si el usuario selecciona la opción "Registrar un nuevo bot", se redirige al usuario a la página que permite dar de alta un nuevo bot en la aplicación (véase Figura A.5). Para iniciar el proceso de dar de alta un nuevo bot el usuario deberá pulsar el botón "Sign in with Twitter" que se muestra en la Figura A.6.

Figura A.6: Registrar un nuevo bot

Tras pulsar dicho botón se redirigirá al usuario a la página de autenticación de OAuth para autorizar la gestión de la cuenta de Twitter que se quiere registrar a través de la aplicación.

Figura A.7: Autenticación OAuth

Para poder continuar es necesario que el usuario haya creado previamente el usuario en Twitter que quiere registrar en la aplicación para que gestione su comportamiento.

Una vez introducidos el nombre de usuario de Twitter y la contraseña que se quieren registrar y tras pulsar el botón "Autorizar la aplicación", se redirigirá de nuevo al usuario a la página principal de la aplicación, pero esta vez se mostrarán los bots registrados por el usuario en dicha aplicación (véase Figura A.8).

Figura A.8: Página principal de la aplicación (II)

A.1.2. Modifica tus bots

Si el usuario selecciona la opción "Modifica tus bots", se redirige al usuario a la página que permite modificar los distintos bots registrados en la aplicación por dicho usuario (véase Figura A.9. En dicha página se permite seleccionar un determinado bot y modificar sus propiedades de configuración.

Figura A.9: Página principal modificación de bots

Tras pulsar dicho botón se redirigirá al usuario a la página principal que permite al usuario modificar los parámetros de configuración de los distintos bots, tal y como se observa en la Figura A.10.

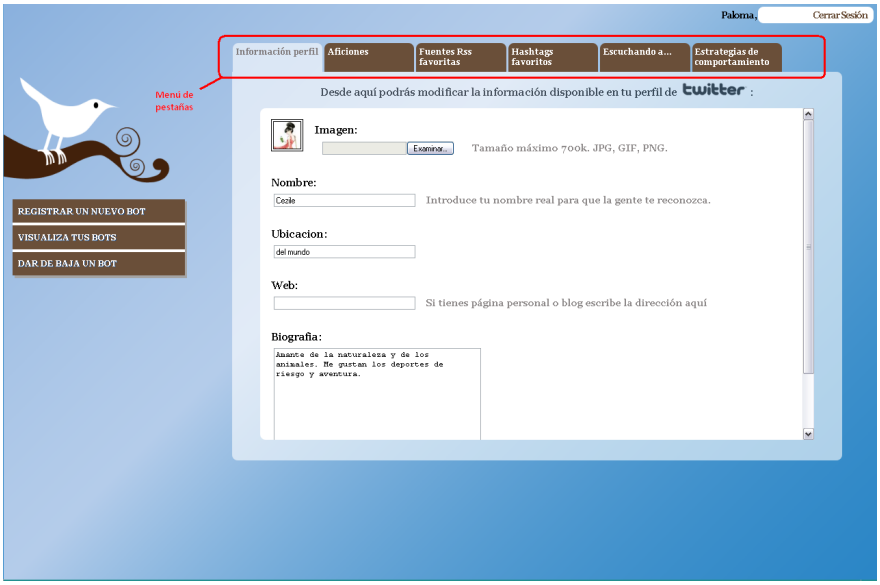


Figura A.10: Menú modificación de bots

Como puede observarse en la Figura A.10, en la página de modificación de propiedades del bot, se muestra un menú de navegación basado en pestañas. Entre las opciones que se ofrecen al usuario se encuentran: *Información perfil*, *Aficiones*, *Fuentes RSS favoritas*, *Hashtags favoritos*, *Escuchando a...* y *Estrategias de comportamiento*.

A continuación se describen detalladamente cada una de las opciones:

Información perfil

Si el usuario selecciona la pestaña "Información perfil" del menú de pestañas se mostrará al usuario la página que permite modificar la información del perfil de Twitter del usuario, esto es, su imagen personal, su nombre real, su ubicación, su página Web y su biografía, tal y como se muestra en la Figura A.11.

Desde esta página el usuario podrá modificar cualquier información del perfil del usuario en Twitter que considere necesarios y tras pulsar el botón "guardar" se enviarán los datos al servidor de la aplicación y se actualizarán automáticamente en Twitter.

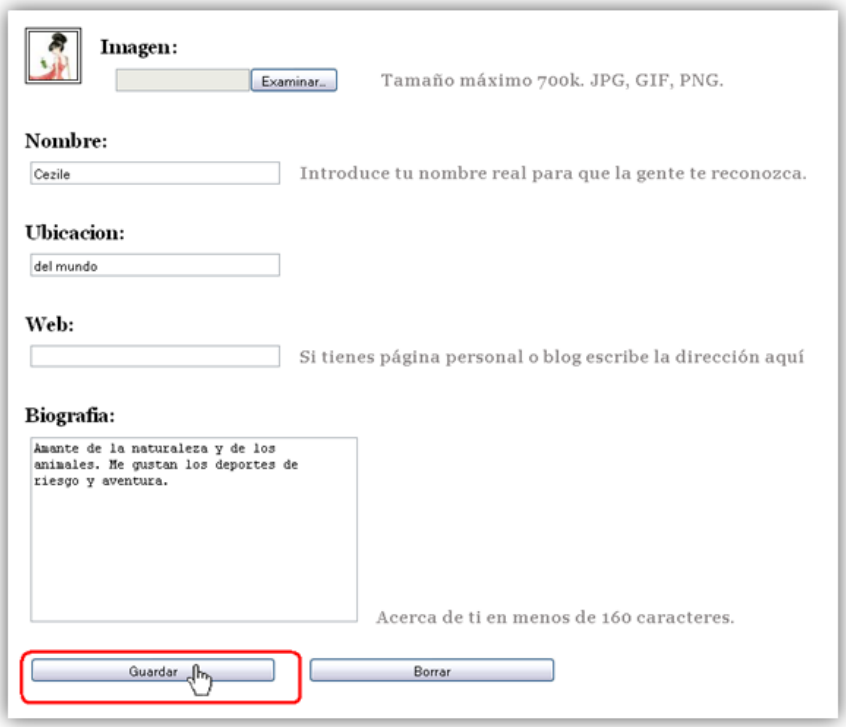


Figura A.11: Pestaña modificación perfil

Aficiones

Si el usuario selecciona la pestaña "Aficiones" del menú de pestañas se mostrará al usuario la página que permite al usuario añadir/eliminar aficiones y asociarla al bot seleccionado (véase Figura A.12).

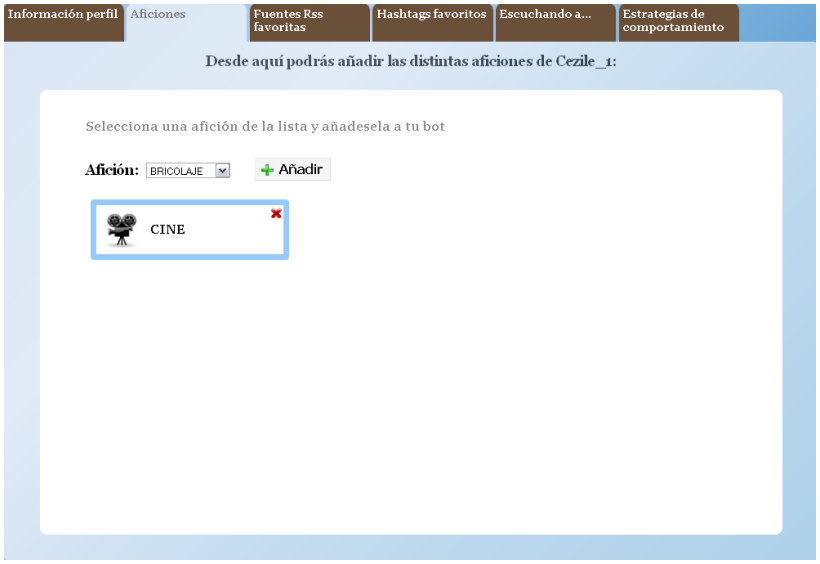


Figura A.12: Pestaña aficiones

Para añadir una nueva afición al bot, el usuario deberá seleccionar una afición de la lista desplegable y pulsar el botón "añadir".

Figura A.13: Añadir afición al bot

Para eliminar una afición asociada a la lista de aficiones del bot, el usuario deberá seleccionar botón con forma de aspa roja que aparece en la esquina superior derecha de la afición que desee borrar de la lista. Tras mostrar un mensaje de confirmación al usuario se borrará definitivamente la afición de la lista de aficiones.

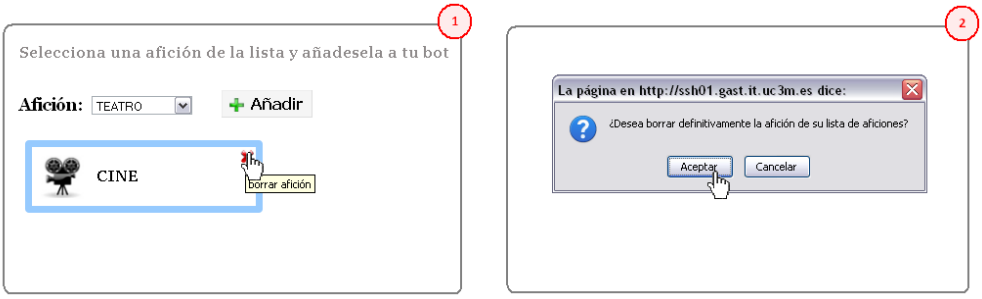


Figura A.14: Eliminar afición del bot

Fuentes RSS favoritas

Si el usuario selecciona la pestaña "Fuentes Rss favoritas" del menú de pestañas se mostrará al usuario la página que permite al usuario modificar la lista de fuentes RSS favoritas del bot seleccionado (véase Figura).

Figura A.15: Pestaña de fuentes RSS favoritas

Tal y como se observa en la Figura A.15, en dicha pestaña se muestran en forma de tabla la lista de fuentes favoritas del bot. Por cada fila se muestra: la URL y el título

de la fuente Rss, la categoría seleccionada por el usuario, el Hashtag (lista de Hashtags) asociados, un botón de borrar (que permite eliminar de la lista la fuente seleccionada) y un botón "añadir a lista general". Este último botón permite añadir la fuente seleccionada a la lista de fuentes RSS general de la aplicación. Como puede observarse en la Figura , este botón puede presentar dos aspectos, tal y como se observa en la Figura A.16. Únicamente cuando el botón tiene el aspecto de la opción (B) actúa de botón, y entonces permite añadir la fuente seleccionada a la lista de fuentes generales de la aplicación Web.

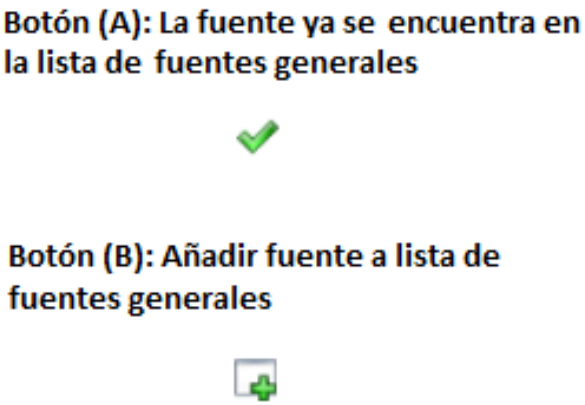


Figura A.16: Botón: Añadir fuente a lista

Para añadir una nueva fuente a la lista de fuentes del bot se deberá incluir la URL de la fuente que se desee añadir, seleccionar una categoría de la lista de categorías y pulsar el botón "añadir".

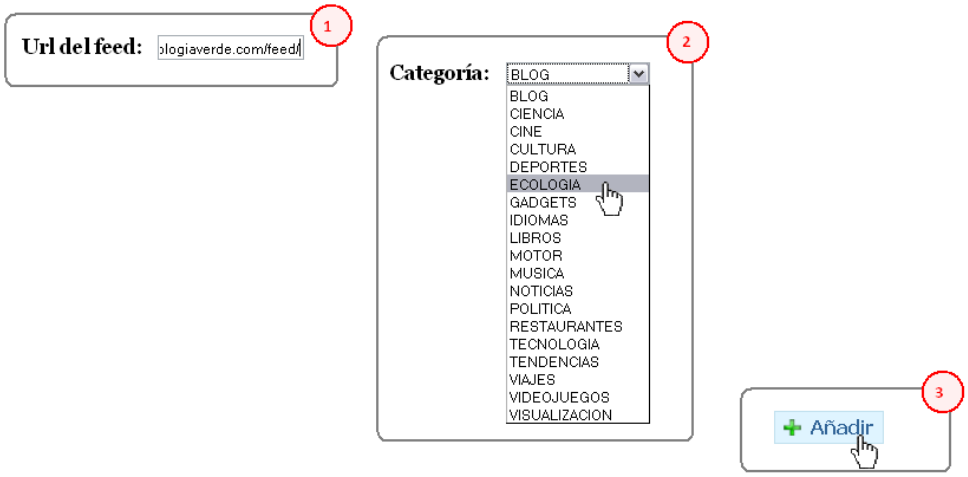


Figura A.17: Añadir fuente a bot

Si el usuario no conoce ninguna fuente, se proporciona la opción al usuario de añadir una fuente de la lista general de fuentes de la aplicación. Para ello, el usuario deberá pulsar el botón "Ver fuentes recomendadas" que aparece en la zona superior de la vista.

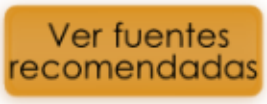


Figura A.18: Botón: Ver fuentes recomendadas

Tras pulsar dicho botón se dirigirá al usuario a la página que muestra la lista general de fuentes RSS disponibles en la aplicación (véase Figura A.21).

Rss	Titulo	Categoria	Añadir
http://www.elpais.es/rss.html	ELPAIS.com - Última Hora	NOTICIAS	✓
http://www.as.com/rss.html	RSS de as.com - Noticias general	NOTICIAS	
http://www.genbeta.com/atom.xml	Genbeta - Software, descargas y novedades	GADGETS	✓
http://elmundo.feedsportal.com/elmundo/rss/portada.xml	Portada // elmundo.es	NOTICIAS	✓
http://www.abc.es/rss/feeds/abc_ultima.xml	RSS Agregación ABC	NOTICIAS	
http://rss.feedsportal.com/c/805/f/535764/index.rss	Público.es - Noticias	NOTICIAS	
http://feeds.feedburner.com/Estrenosblogdelabutacanet	LaButaca.net | Estrenos de cine	CINE	
http://www.guiamp3.com/rss/novedades.xml	Discos, listado de álbumes, todas las novedades - GuiaMP3.com	MUSICA	
http://20minutos.feedsportal.com/c/32489/f/478290/index.rss	Música	MUSICA	✓
http://blogs.wsj.com/numbersguy/feed/	The Numbers Guy	BLOG	
http://datalligence.blogspot.com/feeds/posts/default	Datalligence	VISUALIZACION	
http://analyticsbhups.blogspot.com/feeds/posts/default	Business Analytics	VISUALIZACION	
http://feeds.feedburner.com/MineThatData?format=xml	Kevin Hillstrom: MineThatData	VISUALIZACION	
http://neoformix.com/index.xml	Neoformix	VISUALIZACION	
http://www.springerlink.com/content/1384-5810/preprint/?export=rss	Data Mining and Knowledge Discovery (Online First™)	VISUALIZACION	
http://feeds.feedburner.com/RecordedFutureBlog?format=xml	Recorded Future Blog	VISUALIZACION	
http://feeds.feedburner.com/120linux/post?format=xml	120% Linux	TECNOLOGIA	
http://feeds.feedburner.com/LinuxZone?format=xml	Linux Zone	TECNOLOGIA	
http://feeds.feedburner.com/ilinuxes?format=xml	i Linux es	TECNOLOGIA	
http://estaticos.marca.com/rss/futbol_1adivision.xml	Fútbol / 1ª División // marca.com	DEPORTES	
http://feeds.feedburner.com/AndroidEnEspanol?format=xml	and.roid.es	GADGETS	
http://feeds.feedburner.com/elandroidelibre?format=xml	El Androide Libre	GADGETS	

Figura A.19: Lista de fuentes RSS generales

Por cada fila se muestra: la URL y el título de la fuente Rss, la categoría seleccionada por el usuario y un botón "añadir fuente a favoritos". Este último botón permite añadir la fuente seleccionada a la lista de fuentes RSS del bot. Como puede observarse en la Figura

A.20, este botón puede presentar dos aspectos. Únicamente cuando el botón tiene el aspecto (B) que se observa en la Figura A.20 actúa de botón, y entonces permite añadir la fuente seleccionada a la lista de fuentes del bot.

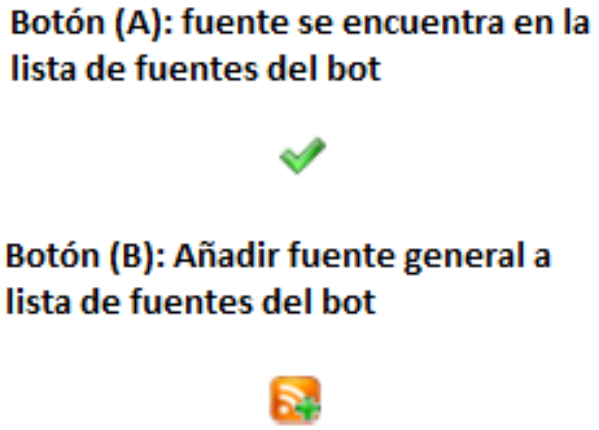


Figura A.20: Botón: añadir fuente a favoritos

Tal y como se observa en la Figura A.21, para una mejor visualización de la lista de fuentes, se permite al usuario seleccionar una categoría de la lista de categorías disponibles y realizar un filtrado de la información disponible.

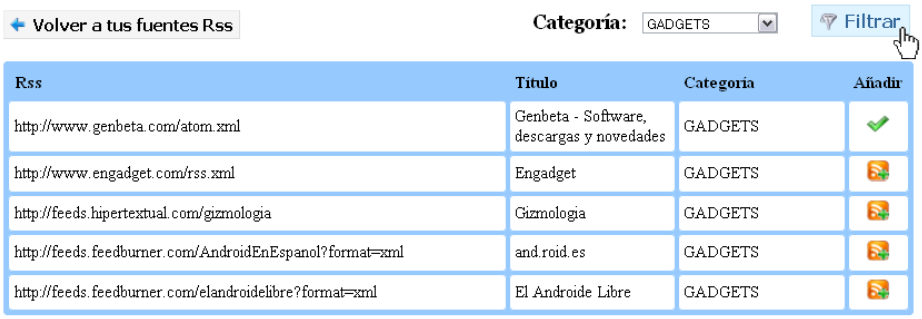


Figura A.21: Filtrado de la lista de fuentes RSS generales

Para volver a la página que muestra la lista de fuentes favoritas del bot, el usuario deberá pulsar el botón “Volver a tus fuentes RSS”.

Por otro lado, para eliminar una fuente de la lista de fuentes del bot se deberá seleccionar la fuente que se desee eliminar y pulsar el botón “borrar”. Antes de eliminar definitivamente la fuente de la lista se pedirá confirmación por parte del usuario.

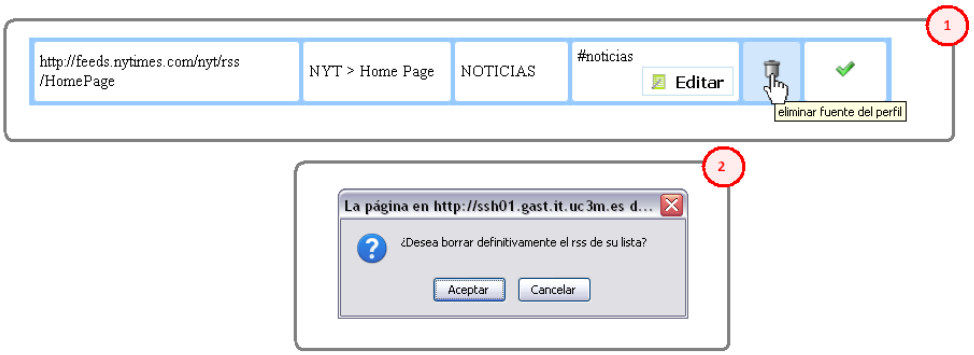


Figura A.22: borrar fuente del bot

Hashtags favoritos

Si el usuario selecciona la pestaña "Hashtags favoritos" del menú de pestañas se mostrará al usuario la página que permite al usuario añadir/borrar un hashtag de la lista de Hahtags favoritos del bot seleccionado (véase Figura A.23).

Figura A.23: Pestaña Hashtags favoritos

Tal y como se observa en la Figura A.23, en dicha pestaña se muestran en forma de tabla la lista de Hashtags favoritos del bot. Por cada fila se muestra: un hashtag, la fecha en la que se añadió, y un botón de borrar, que permite eliminar de la lista el hashtag seleccionado.

Para añadir un nuevo hashtag a la lista de hashtags del bot se deberá incluir en el formulario situado en la parte superior un hashtag, o una lista de hashtags separados por coma y pulsar el botón "añadir".

Figura A.24: Añadir hashtags a bot

Para eliminar un hashtag de la lista de hashtags favoritos del bot se deberá seleccionar el hashtag que se desee eliminar y pulsar el botón "borrar". Antes de eliminar definitivamente el hashtag de la lista se pedirá confirmación por parte del usuario.

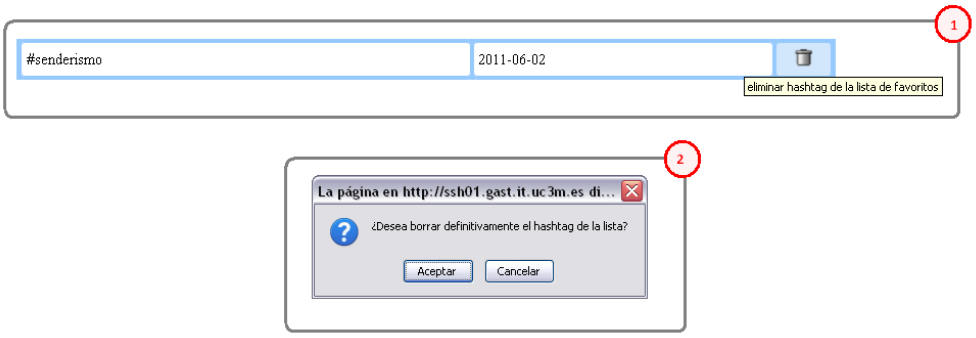


Figura A.25: Eliminar hashtag de bot

Escuchando a

Si el usuario selecciona la pestaña "Escuchando a..." del menú de pestañas se mostrará al usuario la página que permite al usuario añadir/eliminar un Following de la lista de Following del bot seleccionado (véase Figura A.26).

Figura A.26: Pestaña Escuchando a

Para añadir un nuevo usuario a la lista de Following del bot se deberá incluir en el formulario situado en la parte superior el nombre en Twitter del usuario que se desea añadir (con o sin arroba @ delante) y pulsar el botón "buscar".

Figura A.27: Añadir Following a bot

Si se ha introducido un nombre válido, a continuación se mostrará la información recogida del usuario y se dará la opción al usuario de comenzarle a seguir en Twitter., tal y como se muestra en la Figura A.28.

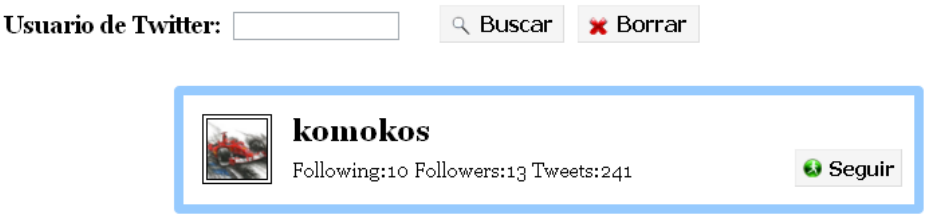


Figura A.28: Seguir a usuario en Twitter

Si el usuario desea que el bot comience a seguir al usuario introducido en Twitter, deberá pulsar el botón "Seguir" y tras pedir confirmación al usuario se enviará automáticamente la petición a Twitter para comenzar a seguir a dicho usuario.

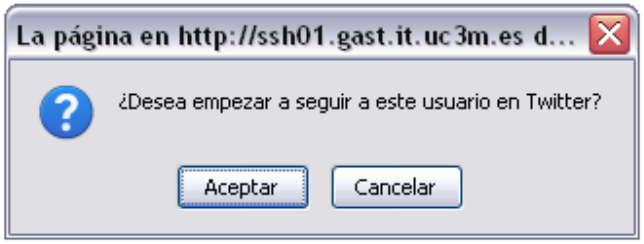


Figura A.29: Confirmación petición se seguimiento

Si el usuario desea que el bot deje de seguir a un determinado usuario en Twitter, deberá seleccionar el usuario de la lista de Following y pulsar el botón "borrar a este usuario y dejar de seguirle".

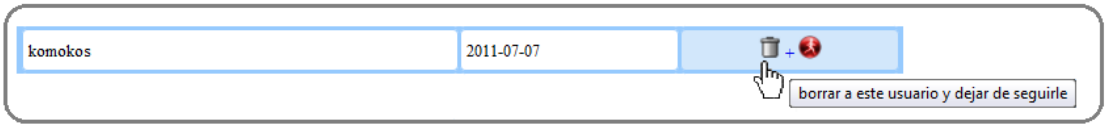


Figura A.30: Dejar de seguir a un usuario

Antes de dejar de seguir al usuario y eliminarlo definitivamente de la base de datos, se pedirá confirmación al usuario. Tras recibir la confirmarción se enviará a Twitter la petición para dejar de seguir al usuario y se eliminará de la base de datos.

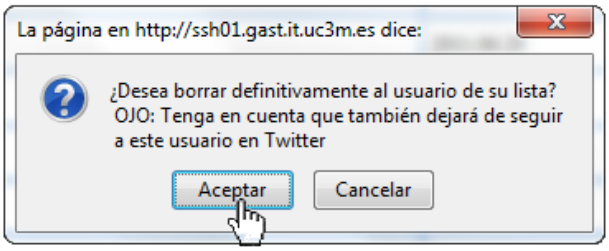


Figura A.31: Confirmación dejar se seguir a un usuario

Estrategias de comportamiento

Si el usuario selecciona la pestaña "Estrategias de comportamiento" del menú de pestañas se mostrará al usuario la página que permite al usuario visualizar la configuración de las estrategias de comportamiento del bot seleccionado (véase Figura A.32).

Figura A.32: Visualización de estrategias

Tal y como se observa en la Figura A.32, dicha página muestra las distintas estrategias de configuración disponibles en menús desplegables: si el usuario desea visualizar la configuración de una determinada estrategia deberá seleccionar la flecha que aparece en la cabecera de la misma para desplegar el menú.

Cada estrategia tiene a su vez un botón "MODIFICAR". Si el usuario pulsa dicho botón se redirigirá al usuario a una nueva página que le permitirá modificar los parámetros de configuración de la estrategia seleccionada (véase Figura A.33).

Figura A.33: Modificación estrategias

Las páginas que permiten modificar cada una de las estrategias tendrán un aspecto similar al mostrado en la Figura A.33. Una vez que el usuario haya configurado los parámetros de una estrategia determinada deberá pulsar el botón "Guardar" y se efectuarán los cambios pertinentes. Una vez reestablecidos los parámetros de configuración de la estrategia, se redirigirá al usuario a la página principal de visualización de estrategias (Figura A.32).

Si el usuario desea volver a la página de visualización de estrategias (Figura A.32) sin realizar cambios en la estrategia, bastará con que pulse el botón "Volver al menú".

A.1.3. Dar de baja un bot

Si el usuario selecciona la opción "Dar de baja un bot", se redirige al usuario a la página que permite dar de baja a un bot en la aplicación (véase Figura A.34).

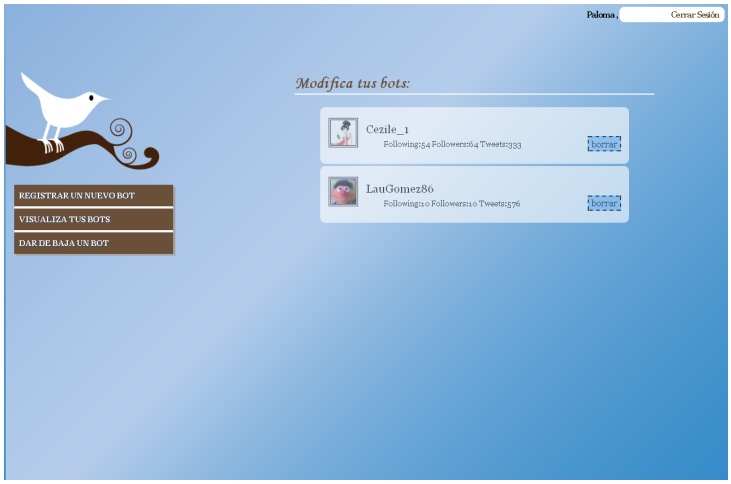


Figura A.34: Página principal dar de baja a un bot

En dicha página se permite seleccionar un determinado bot y darle de baja definitivamente.

Figura A.35: Dar de baja bot

Tras pulsar el botón "borrar" se pedirá una confirmación por parte del usuario. Una vez confirmado, se eliminará definitivamente el bot de la base de datos, junto con toda su información asociada.

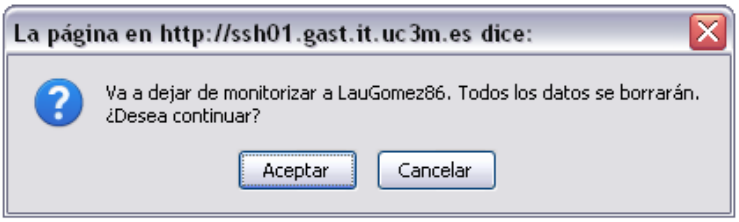


Figura A.36: Confirmación baja bot

Tras haber eliminado los datos del bot de la base de datos, se mostrará una alerta informativa al usuario recordándole que también deberá dar de baja la aplicación desde la página de configuración de Twitter.

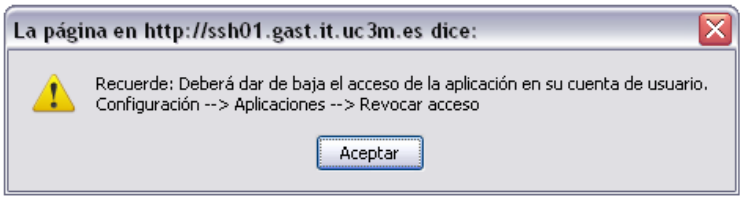


Figura A.37: Alerta baja correcta del bot

Y se redirigirá al usuario a la página principal para dar de baja al bot, donde efectivamente ya no se mostrará el bot que se ha dado de baja.

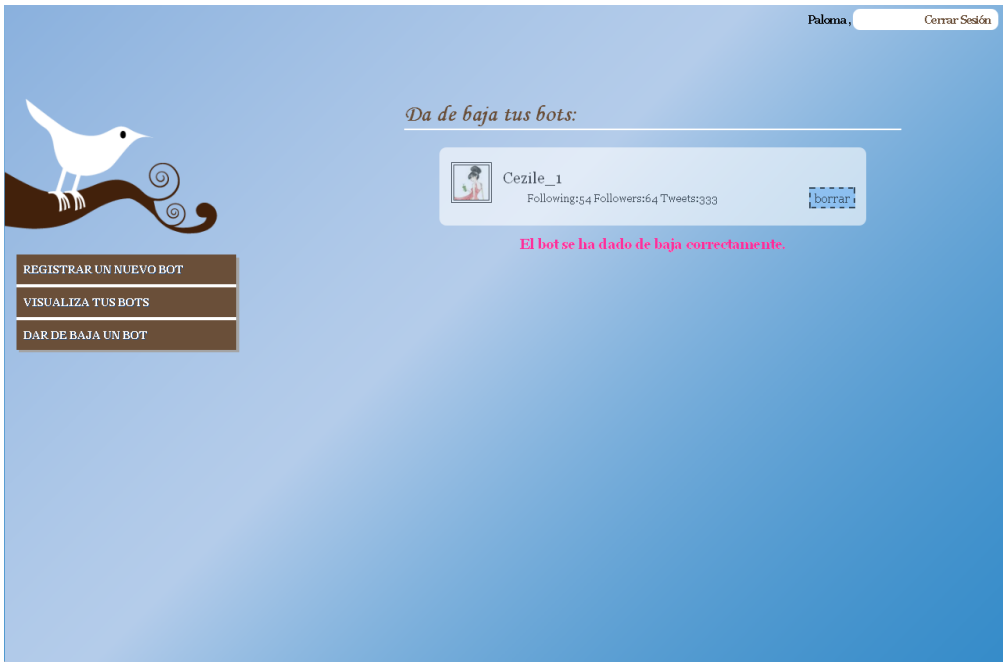


Figura A.38: Página principal dar de baja a un bot (II)

A.1.4. Cerrar sesión

Para salir de la sesión actual, el usuario deberá pulsar el botón "Cerrar sesión" que se observa en la Figura A.5. Tras pulsar dicho botón se cerrará la sesión del usuario y se redirigirá a la página de cierre de sesión que confirma al usuario que se ha finalizado correctamente la sesión.

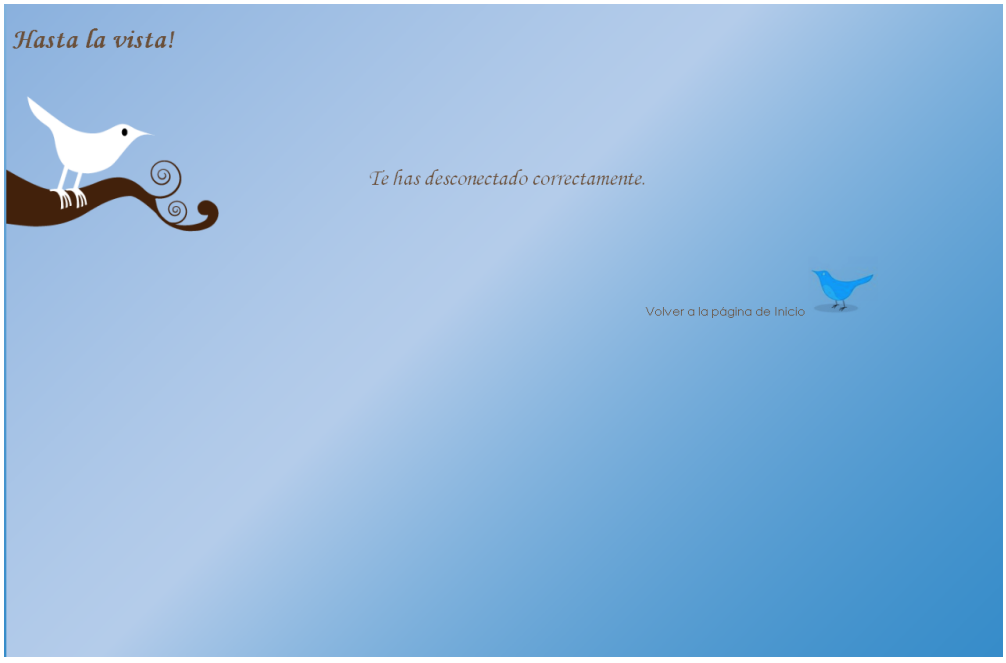


Figura A.39: Cierre de sesión

Apéndice B

Instalación y configuración de Django

Django es un "framework" o entorno de desarrollo de aplicaciones Web basado en el lenguaje de programación Python y que sigue el patrón de diseño MVC (Modelo Vista Controlador).

En este patrón, el "Modelo" hace referencia al acceso a la capa de datos, la "Vista" se refiere a la parte del sistema que selecciona qué mostrar y cómo mostrarlo, y el "Controlador" implica la parte del sistema que decide qué vista usar, dependiendo de la entrada del usuario, accediendo al modelo si es necesario.

La M, V y C se separan en Django de la siguiente manera:

- M, la parte de acceso a la base de datos, es manejada por la capa de la base de datos de Django, la cual se describirá en el capítulo [C.6].

- V, la parte que selecciona qué datos mostrar y cómo mostrarlos, es manejada por las funciones de vista (ver capítulo [C.4]) y las distintas plantillas (ver capítulo [C.7]).

- C, la parte que delega a la vista dependiendo de la entrada del usuario, es manejada por el framework mismo siguiendo la URLconf (ver capítulo [C.5])y llamando a la función de vista apropiada para la URL obtenida.

B.1. Instalación de Django

A continuación se detalla el proceso de instalación del entorno Django.

B.1.1. Requisitos previos de la instalación

Django está escrito en lenguaje Python, por lo que se requiere que Python se encuentre previamente instalado en el equipo. Django funciona con las versiones de Python comprendidas entre la 2.4 y la 2.7. Con versiones posteriores puede presentar imcompatibilidades.

B.1.2. Proceso de instalación

A continuación se enumeran los pasos necesarios para llevar a cabo la instalación de Django:

1. Descargar la última versión disponible de la página oficial de Django [1]
2. Descomprimir el fichero. Una vez descomprimido el fichero, aparecerá un directorio de la forma Django-x.y.z.
3. Abrir un terminal e ingresar en dicho directorio.
   `~$ cd Django-x.y.z`
4. Una vez dentro de dicho directorio, instalar Django.
   `~/Django-x.y.z$ python setup.py install`

B.2. Configuración de Django

A continuación se detalla el proceso de configuración del entorno Django en un servidor Apache con mod_python puesto que es la configuración que se ha utilizado en el proyecto.

Para configurar Django con mod_python al menos es necesario tener una instancia de Apache instalada en el servidor, junto con el módulo mod_python instalado y activado.

```
<Location "/p̃perez/misitio/">
    SetHandler python-program
    PythonHandler django.core.handlers.modpython
    SetEnv DJANGO_SETTINGS_MODULE mysite.settings
    PythonOption django.root /p̃perez/misitio/
    PythonDebug On
    PythonPath "['/home/apt/pperez/lib/micodigo/',
            '/home/apt/pperez/lib/micodigo/misitio'] + sys.path"
</Location>
```

La siguiente configuración indica al módulo mod_python la localización del proyecto Django, es decir, el proyecto se encuentra actualmente instalado bajo el path */p̃perez/misitio/*. Además, si el proyecto creado con Django no se encuentra situado en el PYTHONPATH del servidor, se deberá llamar al módulo mod_python desde la ubicación real del proyecto con la variable **PythonPath**. Este valor deberá contener el directorio padre de todos los módulos importados por la aplicación. En un entorno de producción la variable **PythonDebug** deberá encontrarse desactivada (Off) para no mostrar al usuario ningún mensaje de depuración de Python en caso de producirse cualquier tipo de error.

Por último, será necesario reiniciar Apache, y Apache será capaz de arrancar el proyecto Django [JA08].

[1] `http://www.djangoproject.com/download/`

Apéndice C

Desarrollo de una aplicación con Django

C.1. Crear un nuevo proyecto

El primer paso para desarrollar una nueva aplicación consiste en la creacción de un nuevo proyecto. Un proyecto es un sitio Web completo que consta de una o varias aplicaciones.

Para crear un nuevo proyecto en Django se deberá utilizar el comando ***django-admin.py***. Para empezar a crear el nuevo proyecto, se deberá crear el directorio donde se vaya a trabajar. Dicho directorio no deberá encontrarse en la carpeta raíz del servidor Web (/var/www) por motivos de seguridad, ya que al hacerlo se estaría arriesgando a que la gente sea capaz de ver el código en la Web. Por lo que es recomendable crear dicho directorio **fuera** de la carpeta raíz. Por ejemplo:

```
~$ cd /home/pperez/
~/home/pperez$ mkdir django_proyects
```

Una vez creado dicho directorio, se deberá ingresar en él y ejecutar el comando ***django-admin.py startproject {nombre_del_proyecto}*** en el directorio actual.

```
~/home/pperez$ cd django_proyects
~/home/pperez/django_proyects$ django-admin.py startproject myproyect
```

Una vez ejecutado dicho comando se creará un nuevo directorio con el nombre indicado por el usuario (***myproyect***). Dicho fichero contendrá un conjunto de ficheros **.py**:

```
myproyect/
   __init__.py
   manage.py
   settings.py
   urls.py
```

Cada uno de estos ficheros cumple su cometido para el correcto funcionamiento de Django:

- **__init__.py**: Es un archivo requerido para que Python trate a este directorio como un paquete.
- **manage.py**: Utilidad para gestionar el proyecto: arrancar servidor de pruebas, sincronizar modelos, etc.
- **settings.py**: En él se definen las opciones/configuraciones del proyecto de Django.
- **urls.py**: En este fichero se realizan la declaración de las URL del proyecto de Django. Este fichero sería el controlador de la aplicación. Mapea las URL entrantes a funciones Python definidas en módulos (***views.py***).

C.2. Crear una nueva aplicación

El siguiente paso a realizar, es crear una nueva aplicación web.

Para crear una nueva aplicación dentro del proyecto se deberá ejecutar el comando: ***manage.py startapp {nombre_aplicación}***. Para ello se deberá ingresar en el directorio del proyecto y ejecutar dicho comando:

```
~/home/pperez/django_proyects$ cd myproyect
~/home/pperez/django_proyects/myproyect$ manage.py startapp myapp
```

Una vez ejecutado dicho comando se creará un nuevo directorio (***myapp***) con la siguiente estructura de ficheros:

```
myapp/
   __init__.py
   models.py
   views.py
```

Cada uno de estos ficheros cumple su cometido para el correcto funcionamiento de la aplicación en Django:

- **__init__.py**: Es un archivo requerido para que Python trate a este directorio como un paquete válido.
- **models.py**: En este fichero se definen los modelos u objetos que serán mapeados a una base de datos relacional.
- **views.py**: En este fichero se definen las funciones que van a responder a las URLs entrantes

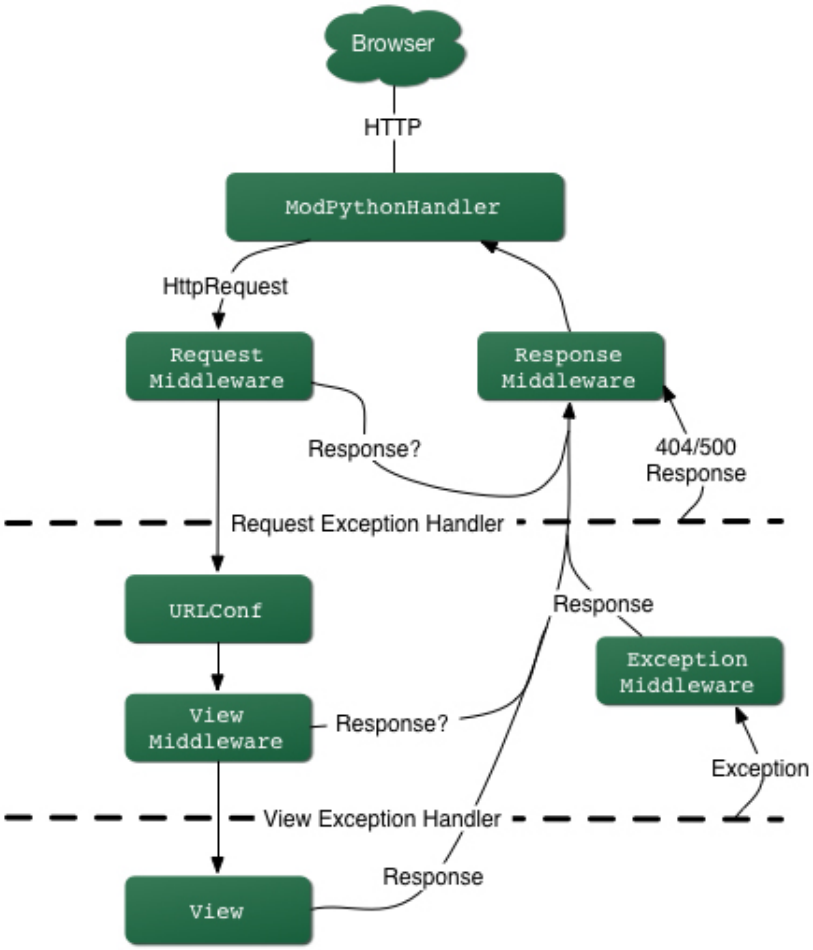
En este punto ya estaría creado el diseño MVC: Modelo (***models.py***), Vista (***views.py***), Controlador (***urls.py***).

C.3. Procesamiento de una petición Django

Para poder empezar con la configuración de una aplicación sencilla, es necesario realizar un breve inciso para comprender cómo Django realiza el procesamiento de una petición.

En la siguiente imagen se muestra el flujo típico –resolución de URLconf a una función de vista que retorna un HttpResponse –del procesamiento de una petición.

Figura C.1: Flujo completo de un petición y una respuesta Django.



Cuando llega una petición HTTP desde el navegador, un manejador específico a cada servidor construye la petición **HttpRequest**. A continuación, el manejador llamará a cualquier middleware[1] de Petición disponible. En el caso de que el manejador retorne un **HttpResponse** se pasará la petición al manejador de Respuesta para finalizar la petición y no se invocará la vista correspondiente.

A continuación Django cargará la URLconf apuntada por la variable `ROOT_URLCONF`. La variable `ROOT_URLCONF` le dice a Django qué módulo de Python debera usar para la URLconf de este sitio Web. El archivo ***settings.py*** que se generó automáticamente durante

[1] Para mayor conocimiento de los middleware de Django se recomienda consultar el siguiente enlace: `http://docs.djangoproject.com/en/1.2/topics/http/middleware/`

la creacción del proyecto, contiene la variable `ROOT_URLCONF` que por defecto apunta al fichero ***urls.py*** de la aplicación.

Una vez cargada la URLconf indicada por la variable `ROOT_URLCONF`, Django comprueba en orden cada uno de los patrones de URL contenidos en la URLconf, comparando la URL solicitada con un patrón a la vez, hasta que encuentra uno que coincida. Cuando encuentra uno que coincide, Django pasa la petición a cualquier middleware de Vista disponible, que se encargará de llamar a la función de vista asociada con dicho patrón. En el caso de que el manejador retorne un **HttpResponse** se pasará la petición al manejador de Respuesta para finalizar la petición y no se invocará la vista correspondiente.

Finalmente, la función de vista será la encargada de devolver el objeto de tipo **HttpResponse**.

En este momento, ya se tienen conocimientos básicos para hacer páginas Web con Django. Es un mecanismo sencillo –únicamente consiste en escribir funciones de vista y relacionarlas con URLs mediante URLconfs utilizando una serie de expresiones regulares. A continuación se verá más en detalle dicho mecanismo.

C.4. Crear una vista

Una función de vista o "vista" es una función de Python que recibe como argumento una petición Web (**HttpRequest**) y devuelve una respuesta Web (**HttpResponse**). Las funciones de vista se declararán en el fichero ***views.py*** que se generó automáticamente al crear una aplicación, aunque se podrán definir en cualquier fichero siempre y cuando se encuentre dentro del Python path [2].

A continuación se muestra la estructura de una función de vista:

```
from django.http import HttpResponse

def nombre_vista (request):
    html = "<html><body>Esto es una página html</body></html>"
    return HttpResponse(html)
```

En dicho código se observa:

- En la cabecera del fichero se deberán importar todas aquellas librerías que sean necesarias para el correcto funcionamiento del código.
- A continuación se definen las funciones de vista. Las funciones de vista toman como primer argumento un objeto HttpRequest, al que típicamente se le asigna el nombre request.
- La primera línea de código dentro de la función, construye la respuesta HTML.

[2] Es la lista de directorios del sistema en la cual Python buscará cuando se use la sentencia *import* de Python

- Por último, la vista retorna un objeto HttpResponse que contiene la respuesta generada. Cada función de vista es responsable de retornar un objeto HttpResponse.

C.5. Mapear URLs a vistas

Una URLconf es un mapeo entre los patrones URL y las funciones de vista que deben ser llamadas por esos patrones URL. Con ello, se le está indicando a Django la función a la que tiene que llamar cuando se escribe una determinada URL de la aplicación en el navegador. Estas funciones de vista deben estar en el Python path.

Para realizar el mapeo se deberá abrir el fichero ***urls.py*** que se generó a la hora de crear el proyecto. Por defecto la configuración de dicho fichero aparecerá como se muestra a continuación:

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    # Example:
    # (r'^mysite/', include('mysite.apps.foo.urls.foo')),
    # Uncomment this for admin:
    # (r'^admin/', include('django.contrib.admin.urls')),
)
```

La variable `urlpatterns` es la misma que Django espera encontrar en el módulo `ROOT_URLCONF`. Esta variable define el mapeo entre las URLs y el código que manejan esas URLs. Por defecto el contenido de esta variable aparece comentado, es decir, la URLconf se encuentra vacía.

A continuación se muestra la forma de editar dicho fichero para que una determinada URL apunte a la vista generada en la sección anterior C.4:

```
from django.conf.urls.defaults import *
from myapp.views import nombre_vista

urlpatterns = patterns('',
    (r'^index/$', nombre_vista ),
)
```

Como se puede observar en el ejemplo, primero se deberá importar la vista que se quiere mapear, *nombre_vista*, en la cabecera del documento. La vista del ejemplo se encuentra en el directorio *myapp/views.py* que en la sintáxis de import de Python se traduce en *myapp.views*.

A continuación se agregará un *URLpattern* que es una tupla de Python en la que el primer elemento es una expresión regular[3] simple, y el segundo elemento es la función de

[3]Las *Expresiones Regulares* (o *regexes*) son la forma compacta de especificar patrones en un texto. Para

vista que se utilizará para dicho patrón. Es decir, de este modo se le indica a Django que cualquier petición a la URL */index* que llegue al servidor se deberá manejar por la función de vista *nombre_vista*.

Este mecanismo de mapeo de URLs a vistas representa un claro ejemplo del principio de acoplamiento débil (*loose coupling*) ya que la decisión de cuál debe ser la URL para una función, y la implementación de la función misma, residen en dos lugares separados. Esto permite al desarrollador realizar cambios en uno de los ficheros sin tener que modificar necesariamente el otro.

C.6. Modelado de datos

Django es apropiado para crear sitios web que manejen una base de datos, ya que incluye una manera fácil y poderosa de realizar consultas a bases de datos utilizando Python. La capa de acceso a base de datos contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es su comportamiento, y las relaciones entre ellos. Actualmente, la plataforma admite tres motores de base de datos: *PostgreSQL*, *SQLite 3* y *MySQL*.

En este proyecto se ha utilizado un servidor de base de datos de tipo MySQL puesto que es el motor de base de datos instalado en los servidores facilitados por la Universidad.

Antes de explorar dicha capa, se deben tener en cuenta algunas configuraciones iniciales, es decir, es necesario indicar a Django el tipo de servidor de base de datos a utilizar y cómo conectarse al mismo. A continuación se exponen los pasos a realizar durante la configuración para que Django utilize un servidor de tipo MySQL.

C.6.1. Configurar el acceso a una Base de Datos MySQL

Antes de realizar dicha configuración, se asume que ya existe un servidor de base de datos MySQL instalado (MySQL 4 o superior), que este se encuentra activado, y que existe creada en él una base de datos (por Ej.: *mybbdd*). También es necesario instalar el adaptador MySQLdb[4] versión 1.2.1p2 o superior, que es una interfaz de Python para MySQL que proporciona las librerías necesarias para que Django interactúe correctamente con la base de datos.

La configuración de la base de datos se encuentra en el archivo de configuración **settings.py** que se creó durante la creación del nuevo proyecto (*myproyect/settings.py*). Por defecto el apartado de la configuración de la base de datos se encuentra vacío.

más información acerca de las expresiones regulares, consultar la página Web `http://www.djangoproject.com/r/python/re-module/`.

[4] `http://www.djangoproject.com/r/python-mysql/`

```
DATABASE_ENGINE = ''
DATABASE_NAME = ''
DATABASE_USER = ''
DATABASE_PASSWORD = ''
DATABASE_HOST = ''
DATABASE_PORT = ''
```

A continuación se describe el comportamiento de cada una de las variables de configuración:

- `DATABASE_ENGINE` le indica a Django el tipo de base de datos que se va a utilizar, en este caso, se le indicará que la base de datos es de tipo MySQL.
- `DATABASE_NAME` le indica a Django el nombre de la base de datos. En el ejemplo *mybbdd*.
- `DATABASE_USER` le indica a Django el nombre de usuario a usar cuando se conecte con la base de datos. Por ejemplo *paloma*.
- `DATABASE_PASSWORD` le indica a Django cual es la contraseña a utilizar cuando se conecte con la base de datos, si se tiene. Si no la base de datos no tiene configurada ninguna contraseña se podrá dejar este campo vacío.
- `DATABASE_HOST` le indica a Django el host dónde se encuentra la base de datos. Si la base de datos se encuentra en el mismo servidor que la instalación de Django, este campo se dejará vacío.
- `DATABASE_PORT` le indica a Django el puerto a usar cuando se conecte a la base de datos. Si se deja este campo vacío, el adaptador de base de datos subyacente usará el puerto por omisión acorde al servidor de base de datos utilizada. MySQL utiliza por defecto el puerto *3306*.

A continuación se muestra un ejemplo de configuración de dicho fichero:

```
DATABASE_ENGINE = 'mysql'
DATABASE_NAME = 'mybbdd'
DATABASE_USER = 'paloma'
DATABASE_PASSWORD = 'xxxxx'
DATABASE_HOST = 'bach.gast.it.uc3m.es'
DATABASE_PORT = '3306'
```

Finalmente, se deberá comprobar que la base de datos se ha configurado correctamente. Para ello, en la consola del sistema se escribirá lo siguiente:

```
~/home/pperez/django_proyects/myproyect$ python manage.py shell
```

Una vez ejecutada la instrucción se entrará en modo shell. A continuación se comprobará la configuración de la base de datos escribiendo:

```
>>> from django.db import connection
>>> cursor = connection.cursor()
```

Si no sucede nada, entonces la base de datos está configurada correctamente. De lo contrario se debe revisar el mensaje de error para obtener un indicio sobre qué es lo que está mal.

C.6.2. Definir modelos en Django

Un modelo de Django es una descripción de los datos en la base de datos, representada como código de Python. Los modelos en Django se definirán en el fichero **models.py** que se creó en el momento de crear la aplicación (*myapp/models.py*).

Cada modelo es representado por una clase Python que es una subclase de `django.db.models.Model`. La clase `Model`, contiene toda la maquinaria necesaria para hacer que estos objetos sean capaces de interactuar con la base de datos y que los modelos sólo sean responsables de definir sus campos, en una sintaxis compacta y agradable.

Cada modelo corresponde a una tabla única de la base de datos, y cada atributo de un modelo corresponde a una columna de dicha tabla. El nombre de atributo corresponde al nombre de columna, y el tipo de campo (por Ej.: `CharField`) corresponde al tipo de columna de la base de datos (por Ej.: `varchar`)

A continuación se muestra un ejemplo de un modelo contenido en el fichero **models.py**:

```
from django.db import models

class Ciudadano(models.Model):
    nombre = models.CharField(maxlength=30)
    direccion = models.CharField(maxlength=50)
    ciudad = models.CharField(maxlength=60)
    pais = models.CharField(maxlength=50)
    website = models.URLField()
```

Es obligatorio que cada modelo disponga de una clave primaria. Si no se indica una clave primaria para el modelo durante su instanciación, Django le asignará un campo de clave primaria entera llamado `id`.

Para obtener mas información sobre los tipos de campo y opciones de sintaxis de modelos disponibles en Django, se recomienda consultar la documentación oficial [5]

[5] `http://docs.djangoproject.com/en/1.2/topics/db/models/`

C.6.3. Instalar un modelo

Instalar un modelo es crear las tablas que representan en la propia base de datos. Para hacer esto, el primer paso es activar el modelo en el proyecto de Django.

Para activar dichos modelos, se deberá editar el fichero de configuración **settings.py** que se creó durante la creacción del proyecto (*myproyect/settings.py*). La variable `INSTALLED_APPS` contenida en este fichero, le indica a Django qué aplicaciones están activadas para un proyecto determinado.

A continuación, se muestra el valor por defecto de dicha variable:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
)
```

Temporalmente se deberán comentar estos cuatro strings añadiendo un caracter (#) al principio cada uno. Son aplicaciones incluidas por omisión porque es frecuente su uso, pero se podrán activar posteriormente si se necesitaran). A continuación, también se deberán comentar las configuraciones por defecto de las variables `MIDDLEWARE_CLASSES` y `TEMPLATE_CONTEXT_PROCESSORS` ya que dependen de algunas de las aplicaciones que se han comentado anteriormente. Entonces, se podrá agregar el modelo creado en el apartado anterior [C.6.2] a la variable `INSTALLED_APPS`. A continuación se muestra el contenido de la variable tras realizar los cambios mencionados (nótese la coma al final del string):

```
MIDDLEWARE_CLASSES = (
#     'django.middleware.common.CommonMiddleware',
#     'django.contrib.sessions.middleware.SessionMiddleware',
#     'django.contrib.auth.middleware.AuthenticationMiddleware',
#     'django.middleware.doc.XViewMiddleware',
)

TEMPLATE_CONTEXT_PROCESSORS = (
#...
)

INSTALLED_APPS = (
#     'django.contrib.auth',
#     'django.contrib.contenttypes',
#     'django.contrib.sessions',
#     'django.contrib.sites',
    'myapp.Ciudadano',
)
```

Cada aplicación en `INSTALLED_APPS` se representa por su ruta Python completa, esto es, la ruta de paquetes separados por puntos que lleva al paquete de la aplicación.

A continuación, una vez que el modelo de datos de la aplicación de Django ha sido activado en el archivo de configuración, se podrá crear la tabla correspondiente en la base de datos. Primero, se tendrá que validar el modelo ejecutando el siguiente comando:

```
~/home/pperez/django_proyects/myproyect$ python manage.py validate
```

Este comando verifica si la sintaxis y la lógica del modelo son correctas. Si todo está bien, aparecerá el mensaje "`0 errors found`". Si no, la salida de error brindará información útil acerca de qué es lo que está mal en el código.

Una vez comprobado que el modelo es válido, se realizará la instalación del modelo en la base de datos. Para ello se deberá utilizar el comando **syncdb**, que consultará la variable de configuración `INSTALLED_APPS` y creará las tablas correspondientes de aquellos modelos que no se encuentren instalados en la base de datos.

```
~/home/pperez/django_proyects/myproyect$ python manage.py syncdb
```

C.6.4. Acceso básico a datos

Django provee una API Python de alto nivel para trabajar con modelos de datos. Para trabajar sobre un modelo de datos en una vista, previamente se importará dicho modelo en la cabecera del fichero de vista. Una vez importado el modelo, se podrán realizar operaciones de insertado, selección, modificacion y borrado de los datos de la base de datos.

Para insertar datos en la base de datos bastará con crear instancias del modelo y llamar a su método *save()*:

```
from myproyect.miapp.models import Ciudadano

def mi_vista (request):
      ...
   c = Ciudadano (
      nombre = 'Paloma',
      direccion = 'C/Madrid, 24',
      ciudad = 'Alcorcón',
      pais = 'España',
      website = 'www.miweb.com')

   c.save()
      ...
```

El hecho de instanciar el modelo no toca la base de datos. Únicamente, cuando se llama al método *save()* del objeto se realiza la sentencia SQL `INSERT` que inserta el objeto en la base de datos. Las subsecuentes llamadas al método *save()* sobre la misma instancia, actualizarán el registro, sin crear uno nuevo (es decir, ejecutarán una sentencia SQL `UPDATE` en lugar de un INSERT).

```
>>> c.nombre = 'Palomita'
>>> c.save()
```

Para seleccionar objetos de la base de datos, se utiliza el atributo *objects*.

1. Para obtener una lista con todos los elementos de un mismo tipo de la base de datos, se utilizará la función *all()* de dicho atributo.
   ```
   >>> lista_ciudadanos = Ciudadano.objects.all()
   ```

2. Para filtrar los datos de una consulta se utilizará la funcion *filter()* de dicho atributo, que devuelve un subconjunto de datos de una lista. Esta función toma como argumentos palabras clave que son traducidos en las cláusulas SQL `WHERE` y `AND` apropiadas.
   ```
   >>> lista_ciudadanos = Ciudadano.objects.filter(nombre = 'Palomita')
   ```

3. Para obtener objetos individuales se utilizará el método *get()* de dicho atributo. Este método devuelve un objeto individual, por este motivo, una consulta de este tipo que devuelva múltiples objetos o que no retorne objeto alguno causará una excepción.
   ```
   >>> c = Ciudadano.objects.get(nombre = 'Paloma')
   ```

Por último, para eliminar una instancia de la base de datos, únicamente se deberá llamar al método *delete()* del objeto. Los borrados son permanentes.

```
>>> c = Ciudadano.objects.get(nombre = 'Paloma')
>>> c.delete()
```

Para obtener más información sobre consultas del modelo de datos, consultar la documentacion oficial de Django [6]

C.7. Sistema de Plantillas

Django posee un potente motor de plantillas que permite separar el diseño de las páginas Web del código fuente subyacente. Una plantilla de Django es una cadena de texto que pretende separar la presentación de un documento de sus datos. Generalmente las plantillas se usan para producir HTML, pero se puede generar con ellas cualquier formato basado en texto.

[6] `http://docs.djangoproject.com/en/1.2/topics/db/queries/`

Una plantilla se compone de variables y etiquetas. A continuación se muestra un ejemplo de plantilla que incluye un código HTML básico:

Cuadro C.1: Ejemplo de plantilla HTML

```
<html>
<head> <title> Título de la página Web </title> </head>

<body>
<p> Esto es una variable {{variable}} </p>

<p> A continuación se muestran unos ejemplos de etiquetas agregadas </p>
<ul>
{% for item in item_list %}
<li> {{ item }} </li>
{% endfor %}
</ul>

{% if condition %}
<p> Esto es una situación condicional </p>
{% endif %}

</body>
</html>
```

Como se puede observar en el ejemplo se tiene que:

- Cualquier texto encerrado por un par de llaves (por Ej.: {{*variable*}}) es una **variable**
- Cualquier texto rodeado por llaves y signos de porcentaje (por Ej.: { % *if condition* %}) es una etiqueta. Una etiqueta indica al sistema de plantillas una orden que tiene que ser ejecutada.

Si se desea obtener más información sobre las etiquetas disponibles en el sistema de plantillas, consultar la documentación oficial de Django [7].

Para usar el sistema de plantillas de Django, se deberán seguir los siguientes pasos:

1. Crear el objeto *Template* y cargarlo especificando la ruta al directorio de plantillas en el sistemas de archivos. Existen otras formas de instanciar una plantilla en Django, pero se ha optado por esta porque es la manera más organizada ylimpia.
2. Realizar la llamada al método *render()* del objeto *Template* con un conjunto de variables, que se denomina *contexto*. Este método retorna una plantilla totalmente renderizada como una cadena de caracteres, con todas las variables y etiquetas de bloques evaluadas de acuerdo a dicho contexto.

A continuación se describen en detalle cada uno de los pasos.

[7] `http://docs.djangoproject.com/en/1.2/topics/templates/`

C.7.1. Cargar un objeto Template

Django provee una práctica y poderosa API D para cargar plantillas del disco, con el objetivo de quitar redundancia durante la carga de plantillas y en las mismas plantillas.

Para usar dicha API, primero es necesario indicar al framework en que lugar del disco se encuentran almacenadas las plantillas, mediante el archivo de configuración (*myproyect/settings*). En este fichero se encuentra la variable `TEMPLATE_DIRS` que es la encargada de indicar al entorno el lugar del disco donde se encuentran almacenadas las plantillas. Por defecto esta variable se encuentra vacía. Una vez decidido el directorio donde se almacenarán las plantillas de la aplicación, se indicará la ruta del mismo en el fichero **settings.py**, tal y como se muestra a continuación (nótese la coma al final del string):

```
TEMPLATE_DIRS = (
    ' /home/myproyect/myapp/templates ',
)
```

A partir de este momento las plantillas de la aplicación se crearán y se almacenarán en dicho directorio.

C.7.2. Renderizar una plantilla

Una vez creado el objeto *Template* el siguiente paso consiste en la renderización del mismo desde la función de vista correspondiente. Para renderizar una plantilla se deberá llamar al método *render()*. Durante la llamada al método se puede indicar un *contexto*.

Un *contexto* es un conjunto de variables y sus valores asociados. Estas variables son utilizadas para rellenar las plantillas y evaluar las etiquetas de bloque. Un *contexto* se representa en Django por la clase `Context`. Su constructor toma un argumento opcional, que consiste en un diccionario que mapea nombres de variables con valores.

```
c = Context ({"nombre": "Paloma" })
```

Puesto que lo más común es cargar una plantilla, rellenar un *Context*, y retornar un objeto *HttpResponse* con el resultado de la plantilla renderizada, Django provee un atajo que permite hacer todo ello en una línea de código mediante la llamada a la función **render_to_response()**, que se encuentra en el módulo **django.shortcuts**. A continuación se muestra un ejemplo de la llamada a dicha función desde una función de vista:

```
from django.shortcuts import render_to_response

def funcion_vista (request):
    name = 'Paloma'
    return render_to_response ('index.html', {'variable': name})
```

Como se puede observar en el ejemplo, la función **render_to_response** recibe dos parámetros:

- `index.html`: que es la plantilla que se está renderizando.
- {'`variable`': name}: que es el contexto. Este parámetro no es obligatorio, si no se introduce, la llamada al método introduce un diccionario vacío.

La llamada al método devuelve un objeto *HttpResponse* con la respuesta ya renderizada. De este modo se incluyen los objetos *Template* en las funciones de vista.

Para más información de opciones avanzadas en el uso y desarrollo de plantillas en Django, consultar la documentación oficial [8].

C.8. Crear formularios en Django

Los formularios en Django se crean de manera similar a los modelos, es decir, mediante clases Python. Por convención los formularios se definen en un nuevo archivo **forms.py** que se creará dentro del directorio de la aplicación (*myapp/forms.py*).

Un formulario de Django es una subclase de `django.forms`. A continuación se muestra un ejemplo de un formulario definido en el archivo **forms.py**:

```
from django import forms

class MyForm (forms.Form):
    comentario = forms.CharField()
    autor = forms.EmailField(required=False)
```

Una vez definidos el/los formulario/s se podrá/n incluir en funciones de vista, tal y como se muestra a continuación:

```
from django.shortcuts import render_to_response
from forms import MyForm

def my_view (request):
    if request.method == 'POST': #Si se ha enviado el formulario...
        #Se recogen los datos enviados en el formulario.
        form = MyForm(request.POST)
        if form.is_valid(): # Si el formulario es válido...
            #Procesamiento de los datos recibidos en el formulario
            comentario = form.cleaned_data['comentario']
            autor = form.cleaned_data['autor']
            return HttpResponseRedirect ('/correct/') #Redirección
```

[8] `http://docs.djangoproject.com/en/1.2/ref/templates/api/`

```
    else:
        form = MyForm() #se crea un nuevo formulario vacío
    return render_to_response ('index.html', {'form': form})
```

En el ejemplo se puede observar que:

- `MyForm(request.POST)` crea un formulario a partir de los datos enviados en la solicitud.
- `MyForm()` crea un formulario vacío.
- la función `is_valid()` procesa los datos enviados en la solicitud para comprobar que sean válidos de acuerdo a una serie de reglas definidas en el momento de crear el formulario (Ej.: el campo *autor* debe ser una dirección de correo electrónico).
- la función `cleaned_data[]` del formulario, convierte el valor recibido en un determinado campo del formulario en un tipo Python válido.

En el ejemplo, en el caso de que los parámetros enviados en el formulario sean inválidos o no se hayan enviado a través de una solicitud, se crea una instancia vacía del formulario y se pasa su valor a través de la variable *Context* a una plantilla (*index.html*). A continuación se muestra el código incluido en la plantilla que mostrará el contenido del objeto de tipo *form* renderizado en la solicitud:

```
<form action="." method="post">
    {{ form.as_p }}
    <input type= "submit" value= "Submit">
</form>
```

`form.as_p` mostrará cada uno de los campos del formulario junto con su etiqueta correspondiente en un parrafo distinto. Tambien se podría usar `form.as_table` para mostrar el formulario en forma de tabla, o `form.as_ul` para mostrarlos en una lista.

Para obtener más información sobre la realización de formularios más avanzados, como por ejemplo, realizar un formulario a partir de un modelo de datos, se recomienda consultar la documentación oficial de Django [9].

[9] `http://docs.djangoproject.com/en/1.2/ref/forms/api/` y `http://docs.djangoproject.com/en/1.2/topics/forms/`

Glosario de términos Twitter

D

Direct Message (DM) Es un mensaje privado entre dos usuarios. Ambos tienen que ser Friends. Estos mensajes se reciben como si fueran correos y se almacenan en la carpeta correspondiente de enviados/recibidos.

F

FF (Follow on Friday) El #FollowFriday o #FF es un hashtag en Twitter que se ha convertido en una costumbre de cada Viernes para la mayoría de los usuarios. Esta costumbre se caracteriza porque al utilizar la clave #FollowFriday (o #FF) en un Tweet se está recomendando seguir a los usuarios incluidos en el mismo.

Follow Es la acción de comenzar a seguir el Timeline de un usuario en Twitter.

Followers Conjunto de usuarios que escuchan el timeline de un determinado usuario en Twitter.

Following Conjunto de usuarios favoritos de un determinado usuario en Twitter. El usuario podrá leer sus tweets en su timeline.

Friends Se dice que dos usuarios son Friends en Twitter cuando existe una relación de correspondencia entre ambos, es decir, ambos leen el timeline del otro.

H

Hashtag Palabras o frases prefijadas del símbolo # que permiten agrupar los distintos tweets de los usuarios por un tema.

L

Listas Es una funcionalidad de Twitter que permite al usuario organizar a sus Followings en diversas categorías creadas por el usuario. El límite es de 20 listas por usuario y de 500 Followings por lista.

M

Menciones (Mentions) Son mensajes de Twitter que incluyen el nombre de un usuario en el cuerpo del mensaje (@usuario), sin importar en qué parte del mensaje está. Todas las personas mencionadas en el mensaje así como los seguidores de la persona que lo emitió podrán ver el mensaje.

R

Reply Es una respuesta al mensaje de otro usuario. Comienza por el carácter @ seguido del nombre del usuario (@usuario) y a continuación el texto del mensaje de respuesta.

ReTweet (RT) Funcionalidad emergente que los usuarios de Twitter han estandarizado de manera "espontánea". Se utiliza para difundir mensajes interesantes vistos en Twitter de forma que se propaguen por las distintas subredes de Twitter. Comienza por el carácter RT seguido por @usuario y el texto del mensaje que se quiere retransmitir.

T

TGIF (Thanks God It's Friday) Es un acrónimo utilizado con frecuencia los viernes entre los usuarios de Twitter, y que demuestra la felicidad de un usuario porque ya es viernes.

Timeline Es un término utilizado en Twitter para describir un conjunto de tweets ordenados cronológicamente y consultados en tiempo real.

Trending topics Son las palabras clave más usadas en un momento dado en Twitter, de lo que más se habla en un momento dado en la herramienta de microblogging.

Tweet (trino, gorjeo) Unidad de publicación en la plataforma Twitter. Mensajes de texto de hasta 140 caracteres.

Apéndice D

Glosario de Términos

N. del T.: API Application Program Interface (Interfaz de programación de aplicaciones)

N. del T.: HTML HyperText Markup Language.

N. del T.: MVC Model-View-Controller.

N. del T.: URL Uniform Resource Locator.

Bibliografía

[Aru10] Gustavo Aruguete. Redes sociales. una propuesta organizacional alternativa. Disponible en la siguiente url: http://practicasgrupales.com.ar//index.php?option=com_content&task=view&id=76&Itemid=, 2010.

[C.G09] Diego C.G. ¿por qué tiene éxito twitter? Disponible en la siguiente url: http://diegocg.blogspot.com/2009/07/por-que-tiene-exito-twitter.html, 2009.

[CGWJ10] Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. Who is tweeting on twitter: human, bot, or cyborg? In *ACSAC*, pages 21–30, 2010.

[CM10] Clair Cain Miller. La historia desconocida de twitter. *The New York Times*, 2010.

[Con09] M. Luz Congosto. Barriblog: Definiendo twitter en pocas palabras. Disponible en la siguiente url: http://www.barriblog.com/index.php/2009/09/30/definiendo-twitter-en-pocas-palabras/, 2009.

[Con10a] M. Luz Congosto. Barriblog: Ficod visto desde twitter. Disponible en la siguiente url: http://www.barriblog.com/index.php/2010/11/30/ficod-visto-desde-twitter/, 2010.

[Con10b] M. Luz Congosto. Lo que siempre quiso saber del api de twitter y nunca se atrevió a preguntar. Disponible en la siguiente url: http://www.barriblog.com/index.php/2010/07/06/lo-que-siempre-quiso-saber-del-api-de-twitter-y-nunca-se-atrevio-a-preguntar/ 2010.

[dadT] Centro de ayuda de Twitter. Siguiendo en twitter: limites y mejores practicas. Disponible en la siguiente url: http://support.twitter.com/articles/100161-siguiendo-en-twitter-limites-y-mejores-practicas/.

[Dur10] Enric Durany. 5 herramientas para monitorizar tendencias en twitter. Disponible en la siguiente url: http://www.enricdurany.com/social-media/monitorizar-tendencias-twitter-trends/, 2010.

[Eli10] Roxana Elizabeth. Articulo sobre redes sociales. Disponible en la siguiente url: http://roxanaelizabeth.wordpress.com/2010/06/19/redes-sociales/, 2010.

[Hil11] Cecilia Hill. Facebook y la historia de las redes sociales en internet. Disponible en la siguiente url: `http://www.tuexperto.com/2011/04/17/facebook-y-la-historia-de-las-redes-sociales-en-internet/`, 2011.

[JA08] Kaplan-Moss J. and Holovaty A. *The Definitive Guide to Django: Web Development Done Right*. Apress, 2008.

[Kov10] Konstantin Kovshenin. Robots are doing better than humans on twitter. Disponible en la siguiente url: `http://kovshenin.com/archives/robots-are-doing-better-than-humans-on-twitter/`, 2010.

[Mar11] Axel Marazzi. Robots sociales fueron programados para engañar twitteros (y lo lograron). Disponible en la siguiente url: `http://gizmologia.com/2011/03/robots-sociales-fueron-programados-para-enganar-twitteros-y-lo-lograron`, 2011.

[MN10] Chih-Hui Lai Mor Naaman, Jeffrey Boase. Is it really about me? message content in social awareness streams. Disponible en la siguiente url: `http://infolab.stanford.edu/~mor/research/naamanCSCW10.pdf`, 2010.

[Nie11] Angela Nielsen. The history of social media [infographic]. Disponible en la siguiente url: `http://www.inspiredm.com/the-history-of-social-media-infographic/`, 2011.

[PRO11] WEB ECOLOGY PROJECT. Help robots take over the internet: The socialbots 2011 competition. Disponible en la siguiente url: `http://www.webecologyproject.org/2011/01/help-robots-take-over-the-internet-the-socialbots-2011-competition/`, 2011.

[Wik11] Wikipedia. Número de dunbar. Disponible en la siguiente url: `http://es.wikipedia.org/wiki/N%C3%BAmero_de_Dunbar`, 2011.